DOCUMENT RESUME

ED 321 667 HE 023 685

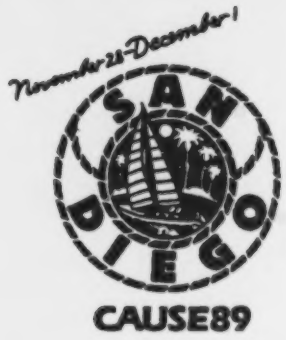
TITLE	Managing Information Technology: Facing the Issues. Track VII: Applications and Technology Issues.
INSTITUTION	CAUSE, Boulder, Colo.
PUB DATE	90
NOTE	79p.; In: Managing Information Technology: Facing the Issues. Proceedings of the 1989 CAUSE National Conference; see HE 023 678.
AVAILABLE FROM	CAUSE Exchange Library, 737 Twenty-Ninth Street, Boulder, CO 80303 (entire proceedings only: $45.00 members, $75.00 non-members.)
PUB TYPE	Speeches/Conference Papers (150) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	Computer Uses in Education; Counseling; Higher Education; *Information Management; Information Retrieval; *Information Systems; Local Area Networks; Organizational Development; Student Records; *Technological Advancement
IDENTIFIERS	Boston College MA; Carnegie Mellon University PA; *CAUSE National Conference; Cuyahoga Community College OH; Louisiana State University; Massachusetts Institute of Technology; Medical College of Wisconsin; University of Virginia

ABSTRACT

Eight papers making up Track VII of the 1989 conference of the Professional Association for the Management of Information Technology in Higher Education (known as CAUSE, an acronym of the association's former name) are presented in this document. The focus of Track VII is on applications and technology issues, and the papers include: "The Effect of Relational Database Technology on Administrative Computing" (Cynthia Golden and Dorit Eisenberger); "On-Line CD-ROM Access in a Digital Environment" (Kent C. Brodie and Carla T. Garnham); "U-VIEW: Student Access to Information Using ATMs" (John J. Springfield); "Implementing a Centralized Directory at LSU" (Emilio A. Icaza and Ouida H. Carroll); "Out of the Blue and into the Black: A Case Study of MIDAS" (Timothy J. McGovern); "A Voice Bulletin Board System for Career Placement" (Norman L. Thienel); "A Counseling Reservation System in a Local Area Network Environment" (Bruce L. Rose); "CASE Tools for the 90s: Deliverance or Extra Burden" (Paul J. Plourde). Each of these papers is preceded by an abstract. (DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED321667



Managing Information Technology: Facing the Issues

Proceedings of the 1989 CAUSE National Conference

TRACK VII: Applications and Technology Issues

November 28 - December 1, 1989
The Sheraton on Harbor Island
San Diego, California

HE023685

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☑ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Copyright© 1990 CAUSE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

CAUSE

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

CAUSE89

Track VII
Applications and Technology Issues

Coordinator:
Michael Naff
Virginia Tech

Many applications have resulted from technological advancements in the past few years, and emerging new technologies promise even more capabilities for enhancing campus information systems. Participants in this track shared information about specific innovative applications from their campuses and the use of newer technologies in higher education. The latter include CASE tools, artificial intelligence/expert systems, hypermedia, object-oriented design, CD-ROM and optical scanning, simulation, teleconferencing, parallel and vector processing, relational data base technology, and the "integrated workstation" concept.

Ouida Carroll, Emilio Icaza
Louisiana State University

John Springfield
Boston College

Bruce Rose
Cuyahoga
Community College

The Effect of Relational Database Technology on Administrative Computing

Cynthia Golden
Assistant Director

Dorit Eisenberger
Senior Programmer/Analyst

Carnegie Mellon University
Pittsburgh, Pennsylvania
November 29, 1989

ABSTRACT

Carnegie Mellon University has chosen to standardize its new administrative system development efforts on relational database and SQL. The INGRES relational database management software is currently in use on several development projects. One administrative computing application, the University Information System (UIS), uses relational database to distribute access to student and employee data. The UIS serves as a good test project for examining how relational database technology has affected administrative computing. The flexibility and the portability of the product were extremely beneficial in our environment. The easy to use 4GL tools allowed us to produce prototypes faster and to quickly make changes to applications. The benefits to the users have been most apparent in the improved access to data that currently exists with the UIS.

Special considerations are associated with moving to relational database technology. Users who want to obtain its maximum benefit need to spend time in learning the structure of their data within the database, the INGRES query tools, and SQL. Administrative Systems also must offer a higher level of support for the new group of people now using the database and develop expertise in the effective management of database applications.

INTRODUCTION

Carnegie Mellon University is an undergraduate and graduate institution located in Pittsburgh, Pennsylvania. Founded in 1905, the former Carnegie Institute of Technology now enrolls 6,900 students and employs approximately 700 faculty and 2,600 staff.

The computing environment at Carnegie Mellon can easily be described as diverse. Students use a variety of personal computers, workstations and some mainframes to do their coursework and research. In the administrative computing environment , word processing and office data processing functions take place primarily on personal computers, particularly Macintosh and IBM models, which are often linked together through local area networks to share resources like printers and file servers. A VAX cluster, (presently consisting of a 6330 , 6230 and three 11/780's), a Sequent Symmetry, a Sequent Balance, and several other workstation-class machines currently support the central administrative computing at Carnegie Mellon. The Administrative Systems (A.S.) department serves the central computing needs of the university.

The stated direction of administrative computing at Carnegie Mellon encompasses the relational database/SQL (Structured Query Language) standard combined with hardware and operating system independence. The INGRES database management software (a product of the INGRES Corporation, Alameda, CA) was selected for use as a basis for all development projects and runs in many operating system environments, including VMS, UNIX, VM and DOS. The operating system and hardware independence will allow us to take advantage of new opportunities presented by the ever-expanding technology base by giving us the freedom to be able to easily move applications from one environment to another.

It is both useful and interesting now to look at the impact of our decision to standardize our development efforts on relational database. Several relational applications have been in use at Carnegie Mellon since approximately 1984. These include systems like Inventory Control, Work Order Management and Telecommunications Management. Recently, a new INGRES-based Human Resource Management System saw its first major release. The Student Information System, which manages student records, admissions and student accounts receivable, was put into production a few short weeks ago. Many other development projects are in progress. We have chosen to examine one system in particular, the University Information System (UIS). This system, which was begun as a project in late 1985, is one of the larger, more widely-used administrative relational database applications at Carnegie Mellon.

THE UNIVERSITY INFORMATION SYSTEM

The UIS is an INGRES application that provides the user with retrieve access to student and/or employee data. The need for such an inquiry system was evident at Carnegie Mellon several years ago, when it was apparent that the central systems, written in COBOL and housed on Digital Equipment Corporation's (DEC) DEC-20's, were not serving the needs of the user community. The central systems allowed access to data in a very limited fashion, and this access was slowed by problems with an over-loaded machine. While plans were being formed to replace these systems, particularly the Student Records and Payroll/Personnel systems, the new versions of the systems were too far away and would not solve immediate problems surrounding access to data. Though some data were available to users in the form of standard screens and printed reports, the need existed for access to data in a more ad-hoc fashion. That is, users wanted to be able to produce their own class rosters, teaching load reports and salary surveys as they were needed or to examine data in their files quickly and easily.

Because the technology to allow users this kind of access to data was available in the form of relational database software, and because the need for access to information on campus was so strong, the decision was made to provide the campus community with an interim solution to its problem. The University Information System was built to give users a way to get the data they needed to do their jobs

while the new systems were being designed and developed. Formatted screens and standard reports, similar to those available in the old system, but more flexible and robust, were included in the UIS, along with the availability of SQL for users to write and run their own ad-hoc queries.

Previous versions of the UIS received their data on a nightly basis, via FTP from the source systems on the old TOPS systems. Now, since the new INGRES-based Student Information System and Human Resource Information System have been released, data contained in the UIS continue to be extracted from the new systems and shipped on a daily basis to a Sequent Symmetry running the Dynix operating system, where they are loaded into an INGRES database. The decision to continue to maintain an "inquiry-only" database after the release of the new systems was made for several reasons. First, having the second database would reduce the load on the primary database. This would be helpful during the first months of the release of the new systems, while bugs were being worked out and the system was monitored. Second, since the feed to the UIS was already in place and the applications were in use, it would not require much effort to simply continue the data transfer. This also allowed the development of the 'inquiry' portions of the new systems to be delayed until a later date.

The UIS data reflect what is contained in the central systems, and are available to users for query access only. Users are not permitted to change any data in the UIS. Any errors found must be corrected at the source, which is the central system.

As mentioned earlier, the UIS data are made up of student and employee information. Separate applications exist for access to these data. Each application consists of standardized screens, queries, reports. A main menu allows the user to choose which subset of the application he wishes to run. The menu, which is written in C, allows the user to choose to run the screens, queries or reports, print copies of the documentation, check when the data were last updated or enter the SQL query facility.

Screens. All the UIS screen applications consist of an entry screen and data screens. The entry screen gives the user the ability to search for a student or employee by name through the "name search" option. A list of all names that match the criteria typed by the user will be displayed in a special screen if the name search option is selected. The user can then select the student or employee he needs and call one of the data screens.

Each data screen in the UIS contains data about one entity: a student or a course or an employee. For example, in the student screens application, the 'roster' screen shows all students in a certain course and section for a given semester.

The user can move easily from screen to screen by typing the new screen code over the current screen code. In the case of a 'roster' screen, this action calls a different screen for the same course without returning to the main menu. The same method applies to changing the entity or the semester being retrieved -- if the user wants to see a new student on a student-based screen like the 'grade' screen, he simply types the id number of the new student over the id number of the current student. If a new semester is desired, the desired semester is typed over the old semester. Any combination of screen code, semester or entity can be changed in order to retrieve new data.

Queries. Commonly-used queries were written and installed in the UIS in a menu format. No special skills, such as knowledge of SQL, are required to run the query option of the application. Users simply select the desired query and enter a few simple parameters. When a query is run, the results are written to the screen as well as to a file, which can be manipulated further by the users.

Reports. The report generation option of the UIS applications gives users the opportunity to produce results for larger, complex queries that are needed on hard-copy, and that are not of a time-critical nature. Again, the desired report is selected from a menu of available reports and parameters are entered prior to execution. In order to control system load during the day and avoid large disk space

allocations for each user that would be required by INGRES to run the reports, the report requests are submitted to a batch queue where they are executed when system utilization is low.

Ad-hoc access. One of the driving forces behind the implementation of the UIS was the desire of the users to have better access to data. The UIS screens, queries and reports were duplications of and enhancements to the features available on the old central systems. The ad-hoc query facility available with INGRES was the key to providing the broader access that many users required in order to be able to retrieve data not available through standard reports or on the inquiry screens. The ad-hoc query option is available from the main menu and requires a knowledge of SQL, the data manipulation language used by INGRES. Users may form their own statements to select data from the database, perform aggregations or do simple file extracts.

The Data. The UIS database consists of approximately 350 megabytes of data representing information from 1975 through the present, contained in over 100 tables or relations. Several types of tables exist in the UIS. They are: main data tables, index tables, translation or utility tables and user-owned tables. The main data tables contain the data shipped to the UIS from the central systems. For example, the Student Schedule Table, which contains a record for every student for every course taken during a semester, consists of almost one million rows 30 bytes wide. Index tables are created when the database administrator defines primary and secondary keys for data access, and are important for efficient execution of queries. Translation tables hold codes and their meanings, and are used consistently by the screens, queries and reports to provide the English translation for a given code. Other utility tables include the tables used for tracking access authorization for users of the system.

Protection. All data in the UIS are considered to be sensitive data (grades, salaries), so tight protection is needed. INGRES provides a permit facility that allows data to be protected at the table level or to be based on values in fields. Also, the days of the week and time of day the database is available to users can also be specified. Once the population is defined for a user with specific permits on data tables, his world of view is limited to that population. Whether he is running the screens, a query, a report or executing his own SQL statement, the data available to him are always that same base population.

User Community. The user community of the UIS application spans a broad segment of the Carnegie Mellon community. Vice-presidents, department heads and administrative and clerical employees from both the academic and administrative sides of the university make up the approximately 200 people who have access to the database.

RELATIONAL TECHNOLOGY

As our current environment has grown from the previous generation of traditional, COBOL-based systems to applications using relational database management software and based on the relational model, we find it useful to look at what advancing to the next generation of computing has brought to the university. The effort involved in making this move is often very clear but sometimes it is difficult to see in advance what the benefits or problems will be.

The advantages of using relational databases for our administrative computing needs are many. We can examine what we see as advantages of relational database from the perspective of the users as well as from the perspective of Administrative Systems.

Advantages to User

Access to Data. For those users who have no need or desire to go beyond the use of simple inquiry screens, any differences between an application written using a relational database manager and an application written in COBOL may go unnoticed. However, those users who in the past were well aware of the structure and limitations of their COBOL application seem to appreciate the additional power they have with direct SQL-based access to the data.

The relational model is easy for novices to understand. The database is structured in the way that one would naturally describe data, grouping data into tables of logically related information about a single entity. As Chris Date, a well known authority on relational database management explains it [1], "a table consists of a row of column headings together with zero or more rows of data values. For a given table, (a) the column heading row specifies one or more columns; (b) each data row contains exactly one value for each of the columns specified in the column heading row." The relationship between tables in the database is expressed by common, key fields. The concept of "data independence" insulates the users, as well as the programmers, from having to understand complex data structures, pointers or underlying data access methods in order for a user to manipulate data. No knowledge of complex data structures, pointers or underlying data access methods is required.

Even the novice user can use the INGRES tools, such as QBF (Query by Forms) or RBF (Report by Forms) to do simple, forms-based inquiries of the data. With a basic understanding of the table structure and SQL, the novice user can perform simple select statements to retrieve data.

For the more sophisticated user, the relational database and its associated tools can be very powerful. With some training in order to become familiar with SQL and learn the database structure associated with an application, users find themselves with the ability to do complex manipulation of data in an ad-hoc manner. Many of our users have created their own tables to use in conjunction with the application tables. These "private" tables can be joined easily with the up-to-date data in the databases. Users have also found it unnecessary to store redundant data in their own personal computer applications, since the central data they need are now easily accessible. It is also much easier now to move any needed data to personal computer applications for use with other software packages, due to the easy-to-use command to extract data from the INGRES databases and the widely available access to data transfer programs like FTP or KERMIT.

Data can be viewed in ways previously impossible without complex programming, largely due to the data independence in relational databases, or the independence of users and user programs from the details of the way the data are stored and accessed. For example, using the old systems to produce a class roster required that a complex COBOL program be run that would produce a roster for every class offered that term. The output file was then divided and distributed to the appropriate departments. Changes to this procedure required the users to request programming and testing by Administrative Systems and often wait weeks for the results. Now, not only can a user do a simple "join" of a few tables, he can limit his request to rosters for a single department or even a single course, all in one SQL "select" statement. Views can also be defined to make life easier for an end-user. A "view" is a way to allow a user to look at one or more tables as one entity, providing logical data independence. When defining the view, the links between the tables are established, and the joins are executed every time the view is selected.

The users' ability to do ad-hoc selections of data on their own has in many cases reduced their dependency on central computing staff. Administrative users on campus need not wait the several weeks it could have taken in the past to have a report they requested written, tested, debugged and run by the central computing staff, who address requests in order of priority. They can write their own request, submit it and have their results almost immediately. The Office of University Planning in particular has made extensive use of the query and report generation facilities available in the University Information System to do the large amount of reporting functions for which they are responsible. Individual departments on campus are also producing their own class rosters, reports and academic audit records with the UIS database.

Participation in Design Process. Perhaps some of the most unique advantages of the relational database for the users concern user participation in the system design process. In more traditional system development environments, user requirements were translated into an information system by the programmers. Often, by the time the system was completed, user requirements would change. Now, because of the easy, rapid prototyping that can be done by the system developers (see next section), users get to see the system in earlier stages in the development life cycle and consequently offer suggestions or criticisms earlier. Users do not have to anticipate all their needed reports or

functions during the system design stage. It is relatively easy to write a new report or add a new screen once the initial system design is in place. Adding such objects does not require any changes to the underlying database and can be done easily using the INGRES tools (see next section.)

Advantages to Administrative Computing

Administrative Systems has indeed seen an increase in productivity and general system quality and flexibility since the relational database and its 4GL tools have been used in systems design and development. The relational model and its way of representing data in tables has been a useful model for developers. In the database design process, the relational model forces the developers and the users to view data in sets and examine the relationship between those sets. This helps to break down the process of designing large, integrated databases into smaller tasks and to facilitate the communication between user and developer.

Productivity. The INGRES software comes with its own tools to build applications, which have helped increase both the level of productivity and the overall effectiveness of the applications programmers in Administrative Systems. The logic portion of the application is specified in ABF (Application by Forms), the application builder. This is separate from screen construction, which is done through VIFRED, INGRES' forms builder, often referred to as a "screen painter". This separation allows for easier and faster changes to be made to an application, since the two functions can be done independently.

By using ViFRED it is simple to specify and later change field location, field validation, screen titles and field display attributes, such as on-screen highlighting, default values or color. Once a screen has been created, any feature can be easily adjusted. The INGRES software handles all the screen I/O, so no complex program is needed. In a matter of a few minutes, a simple screen can be constructed and a user can be entering or selecting data from the database.

Two simple interfaces to the database are available that allow very fast application construction. QBF or Query by Forms, is a screen interface to the database that allows the developer to add, change or delete data. Default QBF forms can be customized with VIFRED in order to add error checking or to customize the screen. No code is required to use QBF in an application. The same kind of function, if written in COBOL, would require hundreds or thousands lines of code. RBF, or Report by Forms, provides simple formatting and retrieval capabilities through a default report on a single table. The default report formats can be changed or enhanced easily. No coding is involved in creating a report with RBF.

A more complex reporting tool, called Report Writer, allows much more complicated reports to be written than does RBF, still with no real code required. Report Writer allows the results of a query to be formatted and aggregated using embedded commands. The Report Writer commands are simple to learn and are combined with the SQL "select" statement to make up the script. A Report Writer script for a very complex report may be only one to one and one-half pages in length. Reports specified with RBF can be written out to a Report Writer script, where they can be changed if a more complicated report is required.

The key to the application building process is INGRES' ABF or Applications by Forms. This tool allows the developer to tie together the pieces of the application written using VIFRED, QBF, Report Writer and 4GL (Fourth Generation Language) code. The 4GL that ABF uses, called OSL, is the tool used to specify most of the application logic.

The ability to write most of the application control and logic in OSL code has in most cases reduced the necessity to write C or COBOL routines. Only in situations where the procedure required was too complex for OSL to efficiently handle or where high performance was so critical was the 4GL application converted to C and optimized for maximum performance.

A simple application can be produced using OSL and the other tools in less than a day. During the application development process, this ability to very quickly make a system prototype has been very valuable to developers. The developer can start with an application "shell" that contains the basic screens and functions, and work with the users to alter or enhance the application. Very little work is required up-front for this process to take place, and consequently little work is lost in designing something the user does not like or wants to change substantially. By using the 4GL application builder, the prototype of the system is the real basis for the final product, i.e., in most cases the prototype becomes the final product.

In general, we have experienced an increase in overall development productivity since the relational database and 4GL tools have been used. The decrease in development time has given us the opportunity to spend more time up-front on systems analysis issues, and has given us the ability to get system prototypes in the hands of the users faster than ever before.

Security. As mentioned earlier in this paper, the underlying permit facility handles access to data at all levels and is very flexible. Administrative Systems' programmers are not required to write data access routines for each application. Facilities for database auditing, checkpointing and journaling have also eliminated the need for any programs to be written to perform these functions important to database security.

Flexibility. We have been pleased with the overall flexibility of the relational databases. In a few cases where they were needed, changes to the system were easy to make, even after the final production release of the application was in place. Adding a field to a table does not affect the applications already in place. The application code will still run, without change or re-compilation. This is a true advantage over more traditional hierarchical or network databases, where application code is based upon the file structure. For example, a major change made to the UIS after its initial release was a re-structuring of the Student Biographical data table. Originally designed to be semester-based, the table contained many fields that were static from semester to semester. The decision was made to change the database to more accurately reflect university record keeping, which would reduce the table size from 800,000 rows to 40,000 rows. This change, which required changes to the data loading programs, some screens, queries and reports, and the database format took less than one week to implement. In a non-relational system, this would have involved a complete re-write and re-compilation of much of the application code.

Portability. In addition to flexibility, portability of applications across different hardware and operating system environments is important to us in order to take advantage of new advances in technology. With the INGRES software we have been able to demonstrate that this is a reasonably simple thing to do. As an example, the UIS was moved from its original home on an IBM 3083 to a VAX 8700 in less than one month, with most of the time being spent on making any operating-system specific changes, such as path names in references to files, in re-compiling COBOL code that had embedded calls to the database and in re-writing the application menu, which was converted from REXX to DCL. When a second port of the UIS was done, the UIS (database only) was moved from the VAX 6330 to a Sequent Balance 8 machine overnight. The pieces of the code written in COBOL and DCL were re-written in C on the Balance, in order to make the application as portable as possible. As a test, we were able to move the entire application from Dynix back to VMS in less than a day.

Maintenance. Development with 4GL's have greatly simplified aspects of system maintenance. Because ABF allows us to tie together many procedures and screens, if a change must be made to one part of the application, the entire application need not be re-compiled. Storing data in tables rather than having it hard-coded in programs also allows users to change applications with no programmer intervention.

Distributed Database Technology. Finally, we see distributed database, in the form of INGRES/STAR, being a true asset to our administrative computing efforts in the years to come. STAR now allows two or more databases to be opened simultaneously, whether they are on the same node or on different nodes. This feature is used frequently in administrative applications when connectivity to other

databases is required, and allows us to eliminate duplication of data, keeping our databases current. A further benefit of distributed database technology is found in the ability to distribute data across nodes which are linked together via a communications network, like DECnet or TCP/IP. The data can be stored on any number of nodes and users at any node can see any of the data without having to know or to specify where it is. In this way, more data are available to more people, while at the same time it is possible to store each piece of data on the machine where it is most often used, in order to maintain efficiency.

OTHER CONSIDERATIONS

Although the benefits of relational database are many, it is also important to look at some of the side-effects associated with moving to this technology and to also look at what is required in order to use relational database effectively and efficiently. Many database vendors make great claims about the rate at which your productivity will increase, how simple the tools are to use and how much less work you will have to do. You are cautioned to not be fooled by such claims. Although productivity does tend to increase, nothing happens overnight!

For Users

Along with the improved access to their own data, users now have the responsibility of learning how to effectively use the tools that are available to them. If they choose not to take advantage of SQL or the other tools like QBF or RBF, most of the advantages of the move to relational database will be invisible to the user. We have found that support of the system by upper management is important to the learning process and the general use of the system. High-level support for the technology has had an effect on its use at Carnegie Mellon, particularly in the case of the UIS. When users at the vice-presidential level used the UIS and saw the advantages that this system afforded, the word spread to others on campus who were also able to take advantage of the newly available data.

It is important to users that they set aside time to attend classes to learn the structure of the database and to learn SQL. An in-depth understanding of what data are contained in which tables is extremely important to the user who will be writing SQL statements to retrieve this data. Some time should be reserved each day for the user to spend in practice sessions in order to become familiar with INGRES and SQL, the database and the operating system. We have seen many users attend an SQL class and then not practice what was covered in class. When they need to do an ad-hoc query, they have forgotten what they had learned, and are unable to make full use of the system.

Although users are required to know more than ever before about how their systems work and may find this learning to be a time-consuming process, it is important to stress to them how the time invested initially in learning the database layout, the application system and SQL will pay off in the future. They will have better access to their own data, will not have to go through a "middle man" (administrative computing) in all cases to get data they need and will be able to respond to their users in a more efficient fashion.

Remember that SQL is still a programming language, and sometimes SQL and database concepts will be difficult for an end-user to master. One should be realistic in these cases, and encourage a user who is having difficulty to master the formatted screen applications first, then attempt SQL use at a later date, once comfortable with the screens. In some cases, either due to lack of initiative, understanding or ability, some users never become comfortable enough with SQL to do their own ad-hoc queries. Many users will be content with the "fill-in-the-form" variety of applications interfaces. [2]

Although some claim that end users are never going to use SQL [3], our experience has shown that a few "key" users arise in each department or administrative area. These individuals tend to become the resident expert in SQL and are most often also the people in the department who have a good understanding of the data. These people have been able to serve as a consultant for others in their department who are experimenting with SQL.

For Administrative Computing

The addition of relational database applications to our computing environment brings with it a new set of responsibilities for Administrative Systems.

Training. Training new users was a significant part of the successful deployment of all our relational database applications. Staff who before were developers and users of the old applications needed to learn how to use the new tools available to them. As the user community grew, many people who previously did not have access to any electronic data also needed to be trained. Our experience with the University Information System showed us that Administrative Systems needed to plan for and present training on the database layout, SQL and other INGRES tools, as well as some operating system concepts. Several sessions, spanning days or weeks are usually required to give most users a basic, working knowledge of the application and the database manager. A few days between training sessions was also found to be helpful. This time lapse gave users some time to practice what was discussed in the formal class, and to return with questions in the following session.

We also had to train our own staff. A few people in the department were relational database "experts" when our major development began, but a good part of our staff were from traditional COBOL or 3rd generation programming language backgrounds. Some training was done internally by our own staff, but we did send large groups to one-week training sessions given by the vendor. In addition, certain staff were sent to specialized vendor training in advanced performance or coding. This all costs money, and you should plan for this in your project budgets. As we have added staff over the past few years, we continue to send them to the one week "INGRES for programmers" course, and will provide internal reviews of database concepts on a regular basis.

Support. In addition to the up-front training requirement, our department must plan for on-going support of these users. It is important to have a staff member available to take "emergency" calls from users who are having trouble with simple tasks like running the screen application, as well as to consult with more sophisticated users who are attempting to write complex queries or Report Writer scripts. Unless help is available, users will sometimes give up on the application.

Access Issues. We have talked some about the benefits of distributing the ad-hoc access to the data with SQL. This method has its drawbacks. In addition to users having problems formulating their queries, there is the danger that they may write and execute "bad" queries. Most often, these are "disjoint" queries that do not properly join tables. The usual consequence of this action is that the database manager executes a Cartesian product and the user's query runs out of disk space and fails. On a few occasions, however, the databases have become inconsistent and consequently inaccessible to all users until a "restore" command can be executed by the database administrator. (Automatic restoration of the database, as well as automatic detection of queries generating very large results are part of version 6 of the INGRES software, but for user of version 5 it is a manual process.)

In addition to the problem with disjoint queries, another unanticipated problem has arisen. Users on occasion are found to be running large, complex queries in interactive mode that are competing with regular production for machine cycles on the central time-sharing systems. No good solution as to how to handle this problem has yet been determined. Not all users can afford to own their own work station, where they could use their own machine resources. Access to the database can be limited to certain hours of the day, or users can be forced to execute all queries in batch mode at a lower priority. This second option would prohibit users from running simple queries during the day. Right now, we are attempting to educate the users who are writing the more sophisticated queries about how their work affects the rest of the users on the machines, and when and how they should do their work. A more permanent solution must be devised.

Many sites have chosen to avoid the problem of users causing inconsistencies in the production databases by providing them with an additional inquiry database. Having a duplicate database for inquiry purposes also provides an additional level of security (no changes can be made to the 'live' database) and helps improve performance for the users of the on-line inquiry screens, who are not

competing against massive update, delete or add transactions. This solution requires that the additional disk space be available, that another database be maintained, and that the additional database be updated on a regular basis. Although this method introduces the possibility of the two databases being "out of sync," depending on the volatility of the data in the source system and the resources available, this may be a good method to use when production databases are involved.

Performance. Getting good performance from a database application is an important part of database management and should be given a high priority. First and foremost, good database design is critical to good performance. A large percentage of the development effort, probably somewhere around 35%, should be spent in the design phase. We have found that when enough time was spent on developing a good design, the system performance issues were easier to resolve.

Once the design is in place and the applications written, if a database is not properly "tuned" the result will be a slow system that will be the source of many complaints. Database programmers must be properly trained in how to take advantage of the database management system and to use it in conjunction with the operating system to its fullest potential. Sufficient time should be allotted during the system testing phase of any project to tune the database and application for maximum performance. The "tuning" process includes properly establishing file structures and secondary indices, distributing the data across disks and gathering statistics used by the internal query optimizer in query execution. Fine tuning should be an ongoing maintenance process. Careful management of all aspects of the database, including indexing, optimization, locking, database integrity and user permits is crucial to the efficient operation of the application.

SUMMARY AND FUTURES

Since the implementation of the first relational database application for administrative use at Carnegie Mellon, a trend has existed toward broader distribution of data and improved access to information. The standardization of all administrative computing on the relational model has brought with it many advantages, both to the user and to the data processing staff. Though some special training and expertise is required to properly maintain, administer and use these systems, the benefits they bring are of critical importance to the operation of this university.

Future directions for relational database in administrative computing see even more work being done in the area of the distributed database environment, through INGRES/STAR, to provide our users with access to many different databases appearing to be a single system. We also see the GATEWAY products, which provide links to other databases (like DEC's RDB) or files, (such as RMS files), to be useful to link existing non-relational systems with INGRES databases.

Finally,we hope to soon be taking greater advantage of the natural language interfaces to databases, which will allow our end users to have full access to their data by simply making their requests in simple English. We hope that this approach will provide users with greater flexibility in their work,and eliminate the requirement of learning SQL to make the greatest use of the data in their systems.

REFERENCES

[1] C. J. Date, *Database: A Primer*, (Addison-Wesley , 1983).

[2] L. Rowe, "Fill-in-the-form programming," In *Proc. 1985 Very Large Data Base Conf.*, Stockholm, Sweden, August. 1985.

[3] M. Stonebraker, "Future Trends in Database Systems," In *IEEE Transaction son Knowledge and Data Engineering*, Vol. 1, No. 1, March 1989.

ON-LINE CD-ROM access in a DIGITAL Environment

Kent C. Brodie
Computer Systems Manager
(brodie@mis.mcw.edu)

Carla T. Garnham
Director
(garnham@mis.mcw.edu)

Management Information Systems
Medical College of Wisconsin
Milwaukee, Wisconsin 53226

ABSTRACT

Even small shops can provide remote access to CD-ROM databases such as MEDLINE, ERIC, AGRICOLA, and Books in Print. Despite having a limited technical support staff, we've done it in a DECNET environment.

Many extremely useful databases are available on CD-ROM. At our medical college, researchers, physicians, and students clamored for CD-ROM hardware and software. This confronted the administration with numerous departmental requests for identical CD-ROM players, microcomputers, and database subscriptions.

Rather than duplicating these systems, we connected a CD-ROM player to a device called a "V-Server" on our VAX system. It isn't magic, but if users' demands are great, funds are limited, and you use the DECNET communications protocol, it works.

This presentation also reviews alternative CD-ROM networking solutions using a Novell network, MACs under TCP/IP, and remote multi-access CD-ROM without networking.

I. Brief introduction to CD-ROM

A. Hardware - how it works.

CD-ROM (compact read-only disk) is a new technology that is breaking all previous barriers for storing computer data. A small 5-inch disk typically holds up to 600 megabytes of data. This is the equivalent of nearly 1,500 magnetic floppy disks of information! CD-ROM stores more because the "focal point" of a laser beam is much smaller than a typical "magnetic" disk head, thus more information can be stored on each disk. This has allowed many companies to "compact" thousands of pages of information on a single disk.

In addition to being efficient, mass-production of CD-ROM disks is very economical. CD-ROM disks are not susceptible to magnetic fields and cannot be accidentally erased.

B. Software - what databases are available?

New titles are being released every day. Typically, CD-ROM applications are based on LARGE database systems. To date, there are over 300 titles available. A sampling of databases (for education) include:

MEDLINE -index to medical journal articles
BOOKS IN PRINT
BOOKS OUT OF PRINT
ELECTRONIC ENCYCLOPEDIA
WORLD ATLAS
ERIC -index to educational research literature
PSYCHLIT - index covering material of psychological relevance scanned from over 1300 journals from more than 50 countries
ABI-INFORM - a business database consisting of abstracts and indexing to business articles from over 800 business and management journals
TOMES PLUS - information on industrial chemicals, hazardous materials, toxicity, and emergency responses.
...and many, many more.

(At this CAUSE session we're distributing copies of the CD-ROM Sourcedisk, a CD which contains CD-ROM product titles and vendors and also copies of the magazine CD-ROM Enduser, which introduces this useful technology.)

C. Cost - Hardware and databases

CD-ROM drives (that attach to a PC) typically cost between $800 and $1400 depending on the type of drive and configuration. CD-ROM disks (databases if you like) can cost as little as $89 or up to $4000. Many CD-ROM databases are purchased as a "subscription" for a yearly fee which usually includes monthly updates. [These costs do not include the microcomputer system.]

D. Why use CD-ROM?

CD-ROM databases are popular because they provide a means to access almost limitless amounts of information in a small space. Depending on your facilities, you may already have most of the hardware needed to implement a CD-ROM database.

The most important factor in considering a CD-ROM purchase is cost. If you need the information, AND if you can implement a CD-ROM system effectively in your institution, then you have cause to consider purchasing a CD-ROM system.

Another factor when considering a CD-ROM purchase is ease of use. There is no standard in the ways software packages are written to access CD-ROM information. Some packages are menu-driven and easy to use; other packages may provide more information, but are more cumbersome. MEDLINE alone is currently sold by more than a half-dozen vendors. Each of these systems is very different from one another. If you get a chance to "demo" a system, do so.

Speed is also a factor. Although CD-ROM disks hold megabytes of information, CD-ROM drives are somewhat slow. Most users are not inhibited by this, as how much information is available usually outweighs how fast they get it.

Nonetheless, there are alternatives to CD-ROM systems, all of which should be considered first.

E. Alternatives to CD-ROM systems

The first alternative that should be considered is "outside access". Many vendors provide access to their databases for a license fee. In the past, we have used the "Grateful MED" dialup service that the National Library of Medicine (NLM) offers. All any user needs is a PC/terminal and a modem. The service is convenient and is offered 24 hours a day.

The cost for this particular service is approximately $23.00/hour for "prime-time" access (9am - 4pm) and $16.00/hour for "non-prime-time" access. Given the number

of researchers at our college that utilize the system, this has proved to be too costly. However, given a different situation, this would be a cost effective way to access MEDLINE.

Another alternative is using "in-house" systems that are NOT CD-ROM based, but normal "load-your-own" systems onto computer disks. Typically, you purchase the software/data that runs on your existing computer system. This is an excellent solution if you can afford it.

II. The problem: A description of our previous non-network setup

A. Hardware configuration: the "standalone" model

Our initial configuration is a "standalone" model. It consists of a PC/AT workstation with two internal CD-ROM drives and a slave printer. This workstation is physically situated in our Medical Library and is exclusively dedicated to running **MEDLINE** (from Online Systems), which consists of a total of eight (8) CD-ROM disks.

Faculty and students use the MEDLINE system to search through literally millions of medical journals to find information relevant to their research. Because of its popularity and effectiveness as a research tool, this workstation is in use almost all day during library hours.

B. Problems and limitations

The primary problem with this setup is that of the location of the workstation. Faculty and students must reserve a time slot to use the system and must physically go to the library to perform their data searches. Our medical complex is physically large, so for many researchers, this proves to be extremely inconvenient and/or impractical.

Additionally, the workstation is only available when the library is open. Many of our faculty perform their research after midnight (or weekend evenings) when the library is closed, and thus they miss being able to use this powerful research tool at their convenience.

Finally, because there are only two disk drives, users occasionally must "swap" disks. This is an inconvenience and slows search time a bit.

C. Cost of adding users - unacceptable!

The cost for a CD-ROM workstation (including all software and licenses) can easily add up to between $5,000 and $8,000. We could attempt to solve the above problems/limitations by "duplicating" this setup across campus in various locations, but the costs would add up very quickly, and not enough staff is available to manage each separate workstation. Obviously, this was not the best way to expand our resources.

III. The solution: A description of our "multi-access" setup

A. The new hardware configuration - uses mostly existing hardware

To solve the problem of purchasing costly redundant systems, and to overcome the fact that PC CD-ROM applications are not usually "networkable", the MIS department used what is called a "V-Server", from Virtual Microsystems, Inc. The V-Server is a device that contains four (4) 286-based processing cells, and lets any VAX terminal (VT compatible) run PC-based applications. At the time, the V-Server was being used by staff who have terminals and who do occasional PC work, usually word processing or spreadsheets. When the need for additional MEDLINE access emerged, we realized that the V-server offered a way to implement remote MEDLINE access.

The V-Server connects to the VAX via a standard ETHERNET cable, and uses the DECNET protocol to communicate with the VAX. Each of the 4 V-Server cells comes with an expansion slot, so it was easy to hook up a CD-ROM disk controller. PC "disks" are actually virtual disks stored on the VAX. The V-Server cells handle the processing, while the VAX handles terminal communication.

[see figure 1 on page 10]

The nicest benefit of this setup for us is that it uses mostly existing hardware: By connecting the CD-ROM disks to the V-Server, MEDLINE instantly became available to all of our hardwired and dialup users.

The problems of physical location of the CD-ROM system have been eliminated, and the cost (approximately $10,000 for a 4-cell V-Server) were definitely reasonable. Note that we also use the V-Server for many other DOS applications in addition to the CD-ROM application, so the money invested was well used.

B. Problems and limitations

Although the V-Server is "PC compatible", there are obviously differences. It took a lot of parameter "tweaking" to install the CD-ROM software properly. We discovered that our MEDLINE uses quite a bit a memory, so we had to upgrade the CD-ROM V-Server cell with expanded memory ($1000) to work properly.

The second problem is that although the V-Server can be accessed by anyone [with a terminal/pc] on our campus, only one person can access the CD-ROM system at a time. This is because only one of the V-Server cells can access the CD-ROM disk. Until recently, this has been a minor inconvenience for our users. However, the CD-ROM system has become so popular that we are addressing the problem of simultaneous, multi-user access to CD-ROM now.

Another problem with the V-Server is, as mentioned above, that it expects a VT-compatible terminal on the user end. The keyboard mapping thus becomes somewhat awkward (i.e., "F6" on a VT220 terminal to enter an "F1" for the PC application). This problem is further compounded by users who use PC's to dial into the system, who end up with a PC emulating a VT terminal emulating a PC. Virtual Microsystems (the V-Server vendor) is currently developing a terminal emulation package for PC users that will map all keys on a 1-to-1 basis to make it much easier to use.

At this time, MACINTOSH users still have to put up with the odd keyboard mapping. Nonetheless, one physician MAC-enthusiast has contrived a way to use the V-Server MEDLINE from his MAC. Once we resolve IBM/PC problems, we will then work on the MAC next.

C. Reaction of staff and students- enthusiastic!

Most of our staff and students were familiar with the Library's CD-ROM MEDLINE system, but many did not use it for reasons described above. When they found out we were hooking up a similar system accessible from the VAX, the reaction was fantastic. We have now set up several users (over 50) who all remotely access the system. The amount of data on the CD-ROM disks, combined with an excellent software package, have helped researchers tremendously.

Our department has received many calls and a several letters commending the system, and they all want more!

D. The next step - Using a NOVELL network in our setup

As mentioned above, a major limitation of our current system is that it can be only accessed by one user at a time. The solution? We might set up the CD-ROM system on some sort of PC-BASED network, such as Novell. However, at first glance, this seems too expensive, and also takes the VAX out of the picture as the communications front-end.

Academic Computing Services at the University of North Carolina (Chapel Hill) has proposed an ingenious setup that uses both solutions, using a Novell network and the V-Server. This solution solves the above mentioned problems.

The setup involves attaching multiple CD-ROM disks to a Novell file server. Each of the V-Server cells is then configured with a network card (in the expansion slot) and thus each V-Server cell becomes part of the CD-ROM network.

[see figure 2 on page 10]

This approach solves our current problems, and is cost effective. A complete network (with workstations) need not be purchased; only one workstation (the server) and up to four (4) network cards. The CD-ROM disks which are now attached directly to the V-Server would instead be attached to the network server.

We have found four vendors of CD-ROM networking products - Artisoft, Inc., Meridian Data, Inc., Online, Inc., and CBIS, Inc. Base systems range from $2,000 for a 5-user system to $15,000 for a 20-user system.

IV. Other multi-user CD-ROM possibilities

A. MACINTOSH-based disk server

The solution(s) shown above all met our needs. We are running a DEC VAX with DECNET, and the V-Server happened to fit into this nicely.

What about non-DECNET sites? In our research of various solutions, we have uncovered some alternatives for sites that may not be configured as ours. One answer involves the use of a MACINTOSH computer as an APPLESHARE file server.

Stanford University recently evaluated a system called "Knowledge Finder" that uses a Macintosh SE computer working as an APPLESHARE file server over a PHONENET network. In turn, this APPLESHARE network is connected to the campus TCP/IP ethernet by using a Kinetics Fastpath gateway.

Thus, users of Macintosh computers across the local PHONENET or other similarly-connected PHONENET networks can all access the CD-ROM server. Their evaluation was successful, and they have implemented the product campus-wide.

Like us, they too implemented a solution that fit into their existing configuration, namely the TCP/IP network.

B. Modification of the Vserver

Another alternative that could be investigated is to have the technical staff at Virtual Microsystems modify the V-Server so that the CD-ROM drive is accessible from ALL of the V-Server "cells", and not just the one cell that the disk controller is attached to.

This gives us a "non-network" multi-user system, and is somewhat cheaper than the Novell system.

This would be a satisfactory solution, but it is slightly limiting. First, we would be required to "invest" a few thousand dollars of funding to have Virtual implement this solution, and it might be quite some time before the finished product is ready.

In addition, this limits our use of the CD-ROM system to the V-Server, and it gives us little flexibility to move to another configuration in the future.

C. Multi-user access WITHOUT the VAX

Obviously, not everyone reading this paper has a VAX. Fortunately, there are many solutions. The latest solution to this problem has only recently been addressed by network vendors. Both Novell and Gandalf Technologies have recently introduced network systems that do not use any front-end whatsoever, nor do they use workstations. Instead, these network systems work with many PC's or ASCII terminals, connected through a serial line. The result is much like a V-Server, in that you can easily provide remote access to networked systems, specifically CD-ROM.

The costs for these systems range from $10,000 to $20,000 (CD-ROM hardware/software not included). This type of system does NOT make use of your existing systems, except for your modems and/or communication lines. However, this type of system does free you from dependance on any particular mainframe system and/or network, and it can offer extremely flexible communications alternatives.

V. Conclusion

A. Evaluate your needs

Should you buy a CD-ROM system? Only your users can tell you. Find out the needs of your user community. Next, compare those needs with your knowledge of what's available in the marketplace. Subscribe to many of the free CD-ROM journals available to get the latest information.

The Medical College of Wisconsin found an increasing need for the MEDLINE database. Since the initial library workstation was installed, our need for access to this type of system has grown exponentially. For us, alternate methods of access (e.g., licensed external services or building in-house disk systems on the mainframe) were too costly to be acceptable.

B. Evaluate your systems

Take a look at your existing hardware. Does it lend itself to hooking up some type of PC-BASED system? There are many options.

DEC VAX systems (like ours) can hook up to many types of computers easily. We happened to have a third party product (V-Server) that made access to PC-BASED applications easy. There are many ways to enable multi-user access to CD-ROM systems. Use your own hardware if you can, as cost effectiveness is always important.

C. Choose the solution

Finally, make the choice appropriate for your institution. We were able to implement a solution that was extremely cost effective with a minimal initial investment. Because we were doing something "new" with our particular MEDLINE CD-ROM software system, the College received a 1-year license grant to make sure the system was functioning properly.

Check with your CD-ROM product vendor to see what type of evaluation periods are available. Although CD-ROM products provide a great way to distribute information, licensing can sometimes be costly, and the wrong choice can mean an unsatisfactory system.

Figure 1. Existing V-Server/CD ROM setup

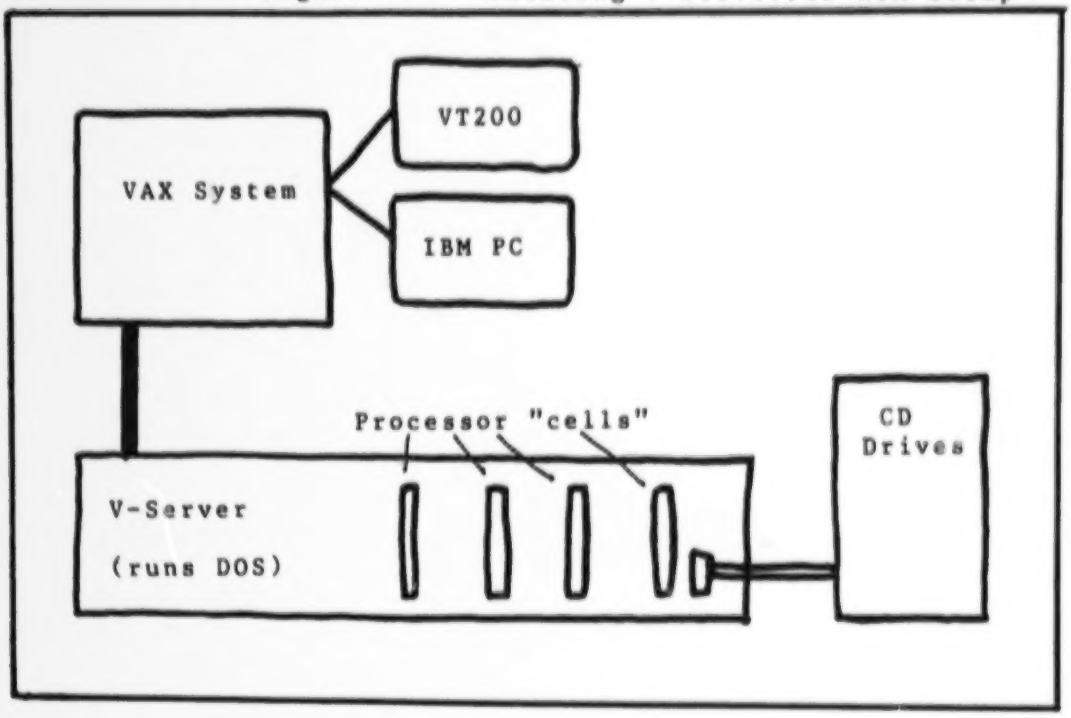
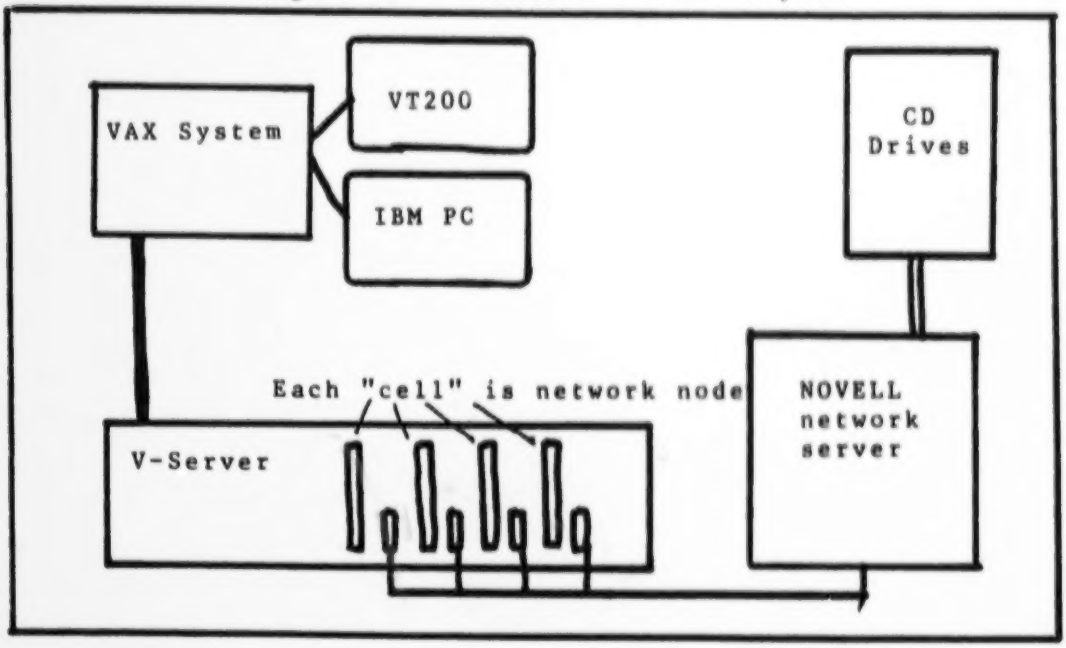


Figure 2. Possible future layout



U-VIEW: STUDENT ACCESS TO INFORMATION USING ATM'S

John J. Springfield
Boston College
Chestnut Hill
Massachusetts

U-VIEW allows students to display and print their Boston College records at automated teller machines (ATM's) located throughout the campus. By inserting a magnetic striped ID card and typing in a personal identification number (PIN), students are allowed access to their courses, grades, student loans, student accounts, financial aid, addresses, and other personal information.

ATM's are "natural" to the student because they can be located in convenient areas and are available after normal office hours. In addition, the U-VIEW ATM's have builtin 80-column printers. U-VIEW has improved life for administrators by eliminating many inquiries. Records are more accurate because students are reporting errors. The ATM security features include the ability to capture obsolete and stolen cards. ATM's are rugged enough to be located in unattended areas. Future uses include the ability to drop and add courses, view urgent messages, and update records.

An Old Idea, A New Idea

The idea of distributing computer access to user departments is not new. However, if we expand the term "user" to include the student (the true "end-user"), we are soon confronted with a whole set of issues that must be confronted. How do we guarantee security? What hours should access be allowed? What kind of devices should be used? Should the devices be located in hallways, dorms, or kiosks? Should full-page printouts be available? Should the new service complement or replace existing departmental access?

After pondering this idea you soon reach the conclusion that you need a device that can read encoded ID cards, is intuitive to use, has a built-in printer, and is rugged enough to resist abuse. In short, you need an ATM (Automated Teller Machine) that has an 80-column printer instead of a cash dispenser.

First Attempt - Terminals with Mag Stripe Readers

Borrowing an idea from David Ridenour of Indiana State University ("Allowing Students Read-Access To Their Own Computer Records", CAUSE/EFFECT ,March 1988), we decided to set up terminals with attached mag stripe readers in three high-traffic locations: outside the registar's office, inside the library, and next to the cashier windows. Since students already had encoded ID cards, we did not have to set up new administrative procedures. By late summer of 1988 students were using "U-VIEW" at these single purpose terminals to access their courses, grades, loans, accounts, financial aid, addresses, and other personal information. U-VIEW was easily used by swiping an ID card through the card reader and supplying a birth date. It was an instant hit with both students and administrators.

However, there were problems with this approach. Specifically:

- Some students inadvertently locked the keyboards by pressing cursor and other keys. A standard refrain from the registrar's office was "Hit the reset key!" Clearly, the standard keyboard was too complicated. All we needed was a simple numeric keypad and a few function keys.

- It became apparent that most students wanted and needed a printout. Long lines were being created because students were copying information from the screen to paper.

- Because the terminals and card readers sometimes confused students, there had to be a staff person in a nearby office to help them. It was obvious that these devices were not self-sufficient enough to be left totally unattended.

- Terminals had to be secured at night because they were not designed to resist abuse or theft.

- Although the system could detect anyone with an old or stolen ID card, we had no way to automatically retain the card.

Experimenting with an ATM

By the fall of 1988 it was clear that we needed a device that looked and acted like the standard ATM that was commonly used by banks. The main difference is that we needed a built-in printer instead of a cash dispenser. Students were used to using bank ATM's on and off campus. If we could find an ATM with a 80-column printer we were sure we could solve our problems. But did such a device exist?

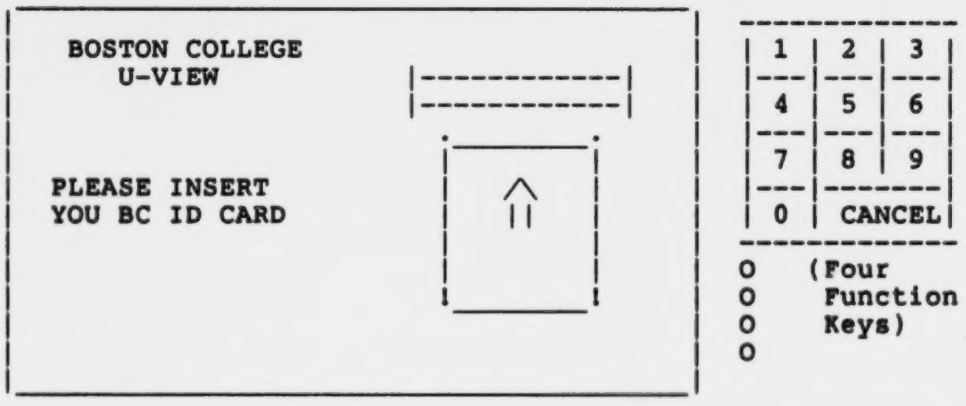
After contacting NCR and Diebold, we discovered that Diebold had a model 1060 "Everywhere Teller Machine". It was exactly what we needed! It had a 20 by 40 column display screen, a numeric keypad, four function keys, and an 80-column printer. But could we communicate with it via VTAM and CICS? It supported SNA/SDLC protocol, but the vendor had not heard of anyone rigging it up in the manner we proposed. After we secured a loaner ATM and manuals from Diebold, we were on our own.

When we received the ATM in December or 1988, our first and biggest task was to see if we could talk to it. Rod Feak, Computer Center Director and seasoned system programmer, dove into the manuals. Without Rod I would have hit a wall. Rod described the ATM to our system as a control unit with a logically attached terminal. To CICS it was set up as a 3600 device. Within a month Rod had CICS talking to the ATM. After that I retrofitted our existing U-VIEW application to work on the Diebold ATM. By February 1989 we went live and became the first college to use an ATM to dispense student information.

U-VIEW on an ATM

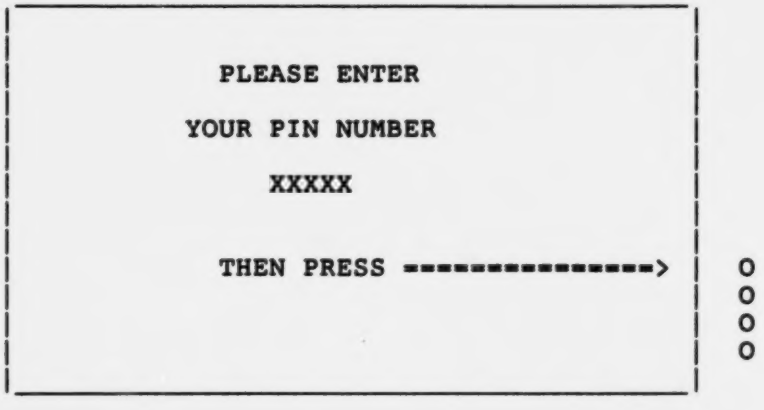
The ATM is left powered up at all times except for periodic maintenance. When it is first powered up, a CICS transaction sends a series of "states" and "screens" to the ATM. These states and screens allow the ATM to have limited functions even when CICS is subsequently brought down. Without accessing CICS the ATM can handle menu navigation, timeouts, and incorrectly inserted cards. But once a student requests data, the ATM sends a message to a CICS transaction and waits for a response. Response time is fast when displaying data. Printing data takes longer because of the relatively slow printer. All access is recorded on a log file on the mainframe.

The first screen appears as follows:



The card is read by the ATM and verified that it is an active BC student ID card. If it has been reported stolen, it is kept by the ATM and reported to the security administrator. It is important to note that all cards are kept by the machine until the student is finished. At the end of a session, the card is partially released to allow the student to retrieve it. If the student walks away, the card is retained by the ATM.

The next screen asks for the student's PIN number:



The PIN is verified. If incorrect, the student is allowed two more tries. The card is rejected after the third incorrect attempt. If the student contnues to reinsert the card and enter an incorrect PIN, the card in retained after 9 tries security administrator is notified.

Once the student is allowed entry to U-VIEW, the main menu screen appears:

```
┌──────────────────────────────────────┐
│                                      │
│            PLEASE MAKE               │
│                                      │
│            A SELECTION               │
│                                      │
│                                      │
│            PERSONAL INFORMATION ====>│  O
│            ACADEMIC INFORMATION ====>│  O
│            FINANCIAL INFORMATION ===>│  O
│            QUIT U-VIEW ============>│  O
└──────────────────────────────────────┘
```

After this menu are various sub-menus and screens showing the desired information. All navigation is done by pressing one of four function keys. Each non-menu screen allows the student to press a function key to print the data on the built-in printer. Since the screen is only 40 characters wide, the printout usually has more detailed information that the screen.

Here is an example of a present semester course screen:

```
┌──────────────────────────────────────┐
│ COURSE NAME          SCHEDULE  LOCATION
│
│ GENERAL CHEMISTRY I   M W F 8  102 CUS
│ GENERAL BIOLOGY I     T TH 10  104 DEV
│ CALCULUS I            M W F 1  207 HIG
│ INTRO TO LITERATURE   T TH 11  001 FUL
│                                      │
│                                      │  O
│                                      │  O
│            PRESS TO PRINT ==========>│  O
│            PRESS TO CONTINUE =======>│  O
└──────────────────────────────────────┘
```

Note that the print and continue functions are always positioned on the last two lines on the non-menu screens. For security reasons screens and printouts never have student ID's or names on them.

By selecting various options at each menu, students are allowed to view and print the following information about themselves.

- Home, local, and parent addresses
- Vehicle parking permits
- Academic status and rank
- Last, present, and next semester courses
- Advisor and registration appointment time
- Student account
- Financial aid
- Student loans

Human Factors

Even though jumping the technological hurdles was personally exciting, it is the human factors that have and continue to be challenging. Students respond well to the simple keyboard and familiarity of an ATM. We tried to mimic the human/machine interaction of a bank ATM wherever possible. However, there are still some human factor problems that do not offer an easy solution.

Our first problem centered around the printing of information. Should we automatically dispense each printout after each selection, or should we wait until the student ends the session? Even though it wastes paper, we found that people want a printout immediately after they press the print function key. So we form feed the paper out of the machine as soon as possible.

Another consideration is the length of time that is allowed to view a screen or answer a question. If we do not allow sufficient time to read all the information (say 25 seconds), we frustrate the user. However if a long response time is allowed, users may walk off without taking their cards!

Our current dilemma involves the changing of the menu structure based upon the time of year. During drop/add period virtual all students want to see their courses. After the end of the semester they want to see their grades. Instead of going through two menus to reach the selection, we could automatically put the most frequent selection on top based upon the time of year. However this may confuse the "frequent user" student who is used to seeing things in the same order.

Shared Administration of U-VIEW

To make U-VIEW work successfully it was imperative that key departments be involved in overseeing it. We were fortunate to already have most of the pieces in place before the project began.

The ID card is issued by the campus police. If a card is stolen of lost the campus police investigate and reissue cards. Cards that are retained by the ATM's are turned over to the police on a daily basis.

MIS handles the programming, the computer center manages and maintains the ATM's, and network services maintains the connections and checks daily for worn-out ribbons and lack of paper.

The security administrator monitors an online log of U-VIEW access. Students are notified if there has been suspicious use of their cards. Statistics are kept on daily and monthly usage.

The registrar and other offices are very helpful in suggesting improvements to present features as well the need for new options.

Acceptance of the ATM

From the beginning the ATM was a success. Everybody seemed to say, "Gee, why didn't we do this years ago?". Students now have "one-stop shopping". Administrators can be freed from answering simple inquiries and use their time on the more involved student questions and problems.

The fall of 1989 showed the following usage of the ATM:

MONTH	NUMBER OF STUDENTS
September	5285
October	2304
November	3828

On slow weekdays about 100 students use the machines. Busy days will show over 500 students.

It should be noted that we expect this count to increase as we add more capabilities. We are purposely keeping the functions static until we have more ATM's. We do not want to simply move the lines from an office to the lines to the ATM's.

Future Enhancements

Currently we have restricted students to inquiry mode only. The next step is to allow updates, viewing and sending messages, and requesting printouts requiring batch processing. Some of this may require an ATM with an alpha keyboard, more function keys, or voice and video. ATM use could also be opened up to faculty, staff, and alumni.

Here are a few ideas for future use of the ATM:

- Drop/Add
  Allow students to drop/add courses that require no departmental permission.

- Student Elections
  Use the ATM like a voting machine. Insure that students only vote once. Voter turnout would increase. Results would be known instantly after polls are closed.

- Updating Local Addresses and Phones
  Students can update their own addresses by choosing from a list of dorms or neighboring streets. A free form address would require an alpha keypad.

- Messaging
  Urgent messages from home, campus police, faculty, adminstrators, or other students could be displayed automatically on the first menu screen. The sending of messages would require an alpha keypad.

- Faculty and Staff Usage
  Allow faculty and staff to view their address and phone, payroll deductions, etc. Currently anyone having access to a CICS terminal can view their records. However some staff may not have access to a terminal.

- Alumni Usage
  Alumni could view their records, request theatre or sports tickets, or request information on current donation projects.

Conclusion

The ATM has proven itself to be an effective way to distribute information to students, free administrators of tedious tasks, and generally improve the quality of life at the university. It's greatest strengths over standard terminals are the security features, convenience, ease of use, resistance to abuse, hours, and ability to print full-page information.

IMPLEMENTING A CENTRALIZED DIRECTORY AT LSU

Emilio A. Icaza and Ouida H. Carroll

Administrative Information Systems
Louisiana State University
Baton Rouge, Louisiana

ABSTRACT

After twelve years of developing online data base systems, LSU achieved a major goal of integrating administrative data processing systems by creating a centralized repository called the Directory. The Directory stores name and address information about an individual, and serves not only as an entry point to such systems as Payroll, Personnel, Student Records, and Traffic, but also as an indicator of an individual's relationship with the University. Because the Directory collects information from administrative offices spanning different areas of responsibility, user coordination was a critical requirement during the development of the system. Both users and analysts were challenged during the design to evaluate the needs of the University as a whole, in addition to the needs of the individual offices. Topics of the presentation include design requirements, special features, and problems encountered during design and implementation.

INTRODUCTION

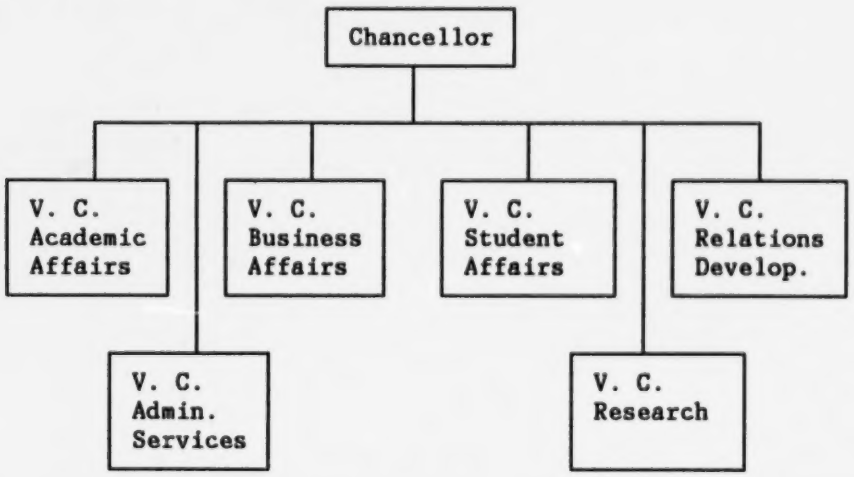
In the course of managing administrative computing development strategies, the direct involvement of users is indispensable. However, users sometimes seem to focus on their immediate needs as opposed to the needs of the University. As a leader of a technology group charged with the responsibility to deliver solutions to the community, what do you do if multi-departmental priorities conflict? How do you consolidate the requirements of many departments into a unified "institutional requirement" and work toward its implementation without direct departmental sponsorship?

In this presentation we will describe the steps that we at Louisiana State University took to design and implement a centralized, University-wide Directory system. We will describe the environment that lead to its development; we will explore the managerial as well as technical issues we faced; and we will give you enough insight into the rewards of doing this so that you will leave here ready to try it yourself.

THE LSU ENVIRONMENT

The University

At the time this project was conceived, Louisiana State University, had a management structure as follows:



Our department, Administrative Information Systems (AIS), reported to the Vice-Chancellor for Administrative Services. For many years, AIS had concentrated its efforts in developing applications that, while focused on the needs of the University as a whole, addressed mainly the needs of the individual sponsoring

departments. Because of this departmental emphasis, a promise that we made in the late seventies, to develop data base systems that shared information, had not yet been fully realized.

For example, offices under all six of the Vice-Chancellors, were managing directory information (name, address, and phone numbers) through the many systems already developed, but they were not all coordinated. Let's visit with some of them to experience the problems they were having.

Public Relations

Meet Libby Paxton. Libby works for the LSU Office of Public Relations. Public Relations reports to the Vice-Chancellor for Alumni Relations and Development. Libby is in charge of Publications. She and her colleagues publish many of the University's publications like the catalogs, brochures, magazines, etc. One such publication is the LSU Phone Book. Like many other universities' directories, the LSU Phone Book includes information about students, employees, and the University organizational structure.

Prior to the Directory system implementation, Libby obtained student name, address, and curriculum information from the Student Records data base. During the fall registration process, students were given an opportunity to correct their name and/or address through scannable forms. After registration, these forms were used to update the data base. Later, the information was extracted, converted from upper case to upper and lower case, and formatted for the typesetting system. While this approach gave the students a chance to request corrections, it was too early in the semester to reflect the many housing changes the students made as they settled down for the school year at the University.

Employee information was kept in a file used exclusively for the Phone Book. Every year, Libby surveyed the LSU campus community to determine if there were changes in employee-department affiliation, name, address, and title. The file was then updated with the results of the survey, and after some validation, formatted for the publishing process.

The University organizational structure and key managerial contacts for each department were maintained in files containing the typesetting markup language codes. To keep these files up-to-date, Libby sent copies of pertinent pages to each department for their review. However, because of the lag between the annual review and the publishing of the book, she had to "monitor the grape vine." For example, organizational changes, and management personnel appointments and promotions approved by the Board of Supervisors, LSU's management board, were incorporated into the files before publication.

The Public Relations environment suffered from some obvious problems. First, the student address information was rarely accurate and in order to improve its quality, the Phone Book was often delayed to allow for updates to the Student Records data base. The name and address data in this data base were kept in all upper case, while the Phone Book was published in upper/lower case. This meant that a considerable programming effort was required in creating the publishing files from the data base, seldom with optimal results. Second, Libby's employee information did not carry the social security number, as a result, it could not be used to update the personnel data base. This deprived the rest of the community from the benefits of Libby's efforts to obtain current information.

Office of Telecommunications

Meet Chip Dodson. Chip is in charge of the Office of Telecommunications and reports to the Vice-Chancellor for Administrative Services. Chip has a group dedicated to provide directory assistance about LSU to the community. This group is supervised by Sandra Hodges. Sandra and her colleagues dispense phone and address information to callers.

Prior to the implementation of the Directory system, Directory Assistance relied on several sources for the information they were asked to provide. The primary source of information was the Phone Book. Sandra kept a "master" for the office with hand-written corrections and additions. She updated the master with changes she got though their daily contacts with the community. Hand-written cards provided by the Personnel Office informed them about new employees and changes in job classification.

In early 1987, Chip's predecessor came to us with the desire to do something about the Directory Assistance service. He wanted to explore new ways to obtain existing information on students and employees and improve the quality of the service. He was facing some turnover due to retirement and felt that the experience level of the operators would be difficult to replace.

Office of The Treasurer
Office of Parking Traffic and Transportation

Meet Judy Williams. Judy works in the Treasurer's office supervising billing and student fee collections. The Treasurer reports to the Vice-Chancellor for Business Affairs. Judy is talking to Gary Graham, who is the director of Parking, Traffic and Transportation. Gary reports to the Vice-Chancellor for Administrative Services, and is in charge of monitoring parking areas, traffic flow, and alternative means of transportation on campus.

Prior to the Directory system implementation, these two offices shared names and addresses which were separate from the student and employee records. Students gave changes to a Student Records clerk who updated the Student Records data base. Employees communicated changes to a personnel officer who updated the employee (Human Resource Management, HRM) data base. This information rarely got back to the Treasurer and Traffic name and address data base. Only through returned bills for traffic tickets, deferred notes, housing charges, etc. would these offices know that the address was incorrect.

Social security numbers, used as the key in most campus systems, were also maintained independently from the rest of the campus data bases. During the registration processing cycle, Judy needed to match fee collections to credit hours to verify student fees. The discrepancies in SSN that had developed during the previous year made this matching process more difficult and time consuming.

The Students

Ellen, Nancy, and George meet in the LSU Student Union between classes. Ellen just got married and claims that her married name is already in her transcript but not on her student paycheck. Nancy is disgusted that her student loan check was sent to her old apartment even though the Phone Book has her new address. George is puzzled that his phone number is wrong in the Phone Book, even though he requested a change at registration.

Before the centralized Directory, students were not aware that informing any University office of changes in name or address did not guarantee these changes would be effective across campus. Students might have had to report a change to five different offices in order to get all records changed. Their frustration, after repeated attempts to rectify the situation, may have caused them to abandon their effort to make the University aware of their location.

THE IMPLEMENTATION

Now that we have described the environment and the problems that were afflicting many of our users, we need to concentrate on the steps we took to implement a solution.

Armed with a request from the Office of Telecommunications to develop a system for their Directory Assistance operators, we embarked on a journey to produce a centralized solution for the University.

LSU's development methodology breaks down the applications development life cycle into Requirements Definition, External

Design, Internal Design, Program Development, and Installation. At this time we were at the end of the Requirements Definition which establishes what needs to be done. But before we could proceed with the definition of the how, the External Design, our Director requested an initial assessment to concentrate on the following questions:

- Could a solution be found that would fit the current systems and those systems currently in development?
- How much effort would be required?
- Could the implementation be staged in such a way so as not to interfere with ongoing development efforts?

A detailed investigation was initiated to identify the primary systems using directory information. Eight out of 18 installed data base systems were managing this type of information: Admissions, Student Records, Independent Study, Human Resource Management, Traffic, Treasurer, Fee Bills, and Housing. Two of three systems in development would also be affected: Financial Aid and Telephone Registration. Fitting a solution that extracted information from that many systems would be a data management nightmare, so we concluded that the best solution was to develop a separate repository of directory information. This new data base should satisfy all requirements currently implemented in existing systems, and also serve as the source of information for the Phone Book and Directory Assistance.

The AIS organization was divided into four groups: Development, Maintenance, Technical Services, and Strategic Systems.
Following the installation of DB2 in 1986, and the success of our pilot project, the development group had a large inventory of systems in development. For this reason, this project was assigned to Strategic Systems.

It was estimated that by using DB2, the major components of the system could be completed in six to eight months. These estimates also told us that about 85 percent of the effort required to develop the system would be changing existing programs to access the new data base. For this reason, and to keep the impact to our current development commitments to a minimum, the participation of the AIS maintenance group was deemed indispensable.

The maintenance group responded with enthusiasm to our proposal. They were excited about doing work with DB2 and agreed with us about the many benefits that a project like this would have on the existing environment.

Now we needed the cooperation of our user community.

User Participation

A meeting was called with user representatives from all departments involved. The agenda for the meeting included:

- A description of the current environment and the inherent problems we had identified.
- A proposal for an institutional name and address data base that would serve as a central source of information for all systems.
- A description of the requirements that had been identified.

Attending this meeting were six directors and ten Associate or Assistant directors reporting to five Vice-Chancellors from two campuses. The outcome of the meeting was positive.

Users were then contacted individually. We wanted to make sure that all of their concerns were addressed. The following issues were identified in the interviews:

- The implementation of the Directory should not significantly alter the screen flow of the systems interfacing with it.
- Social security maintenance procedures should be part of the Directory system implementation. It was suggested that since most systems use SSN as the key to records on individuals, discrepancies in them must be minimized for the Directory system to be successful. Some users felt that SSN changes should be restricted and demanded weekly notifications of SSN changes.
- Availability of SSNs through Directory inquiry should be restricted to "those who need to know" to discourage unauthorized access to sensitive information in the target systems.

Since none of the issues discussed above would seriously impact the implementation of the system, a decision was made to proceed with the project.

The Design of the System

One of the first decisions that was made concerning the design of the system was the establishment of a model to interface with other systems. After some consideration, the Server/Requester model was chosen. With this model, the Directory system would act as a server and all the other systems would be requesters. The requester functions would be implemented in the form of subroutines that can be used by each system to satisfy their data needs. This model would give the system enough flexibility to be able to accommodate most requirements on a global basis instead of a system-by-system basis. However, we knew that we were also committing ourselves to a very demanding development

and maintenance environment. All system needs would have to be satisfied by the Directory subroutines quickly to avoid delays in the implementation schedule.

Once the model was established, our standard Entity-Relationship modeling procedures were followed to establish the data model and to define the contents of the DB2 data base tables (Figure 1.)

The system was designed with the following features:

- Access to information by SSN, spelling of the name, or sound of the name.
- Quick display of name, title, organization, phone, and relationship with the University (i.e. faculty, staff, student, etc.)
- Ability to handle requests for Directory hold so as to satisfy the Buckley Amendment and personal privacy issues.
- Ability to keep a history of changes made to name and social security numbers to aid in the resolution of conflicts.
- Ability to determine what systems carry information about an individual from a central location.
- Ability to support multiple address types, but minimize storage redundancy when several address types share the same information.
- Provide for decentralized maintenance of address data to maximize the chances of capturing correct information at any University office.

At the end of both the External Design and Internal Design, all user representatives were given an opportunity to review the definition of the system. Later in the development cycle, the users were again brought together to discuss pre- and post-implementation procedures. We took every step possible to keep the users aware of our progress with the implementation so as to minimize conflicts.

In the summer of 1988, the Centralized Directory became a reality. By far the most pervasive problems that we faced during the early stages of its implementation were data related. In deciding what information to load into the new data bases, a priority scheme was worked out so that information was loaded from the employee and student data base before any of the other sources. This created some confusion among the users whose data was preempted by a previous load. However, after a few hectic days, the users either changed the data back, or accepted the change, and soon settled down to work.

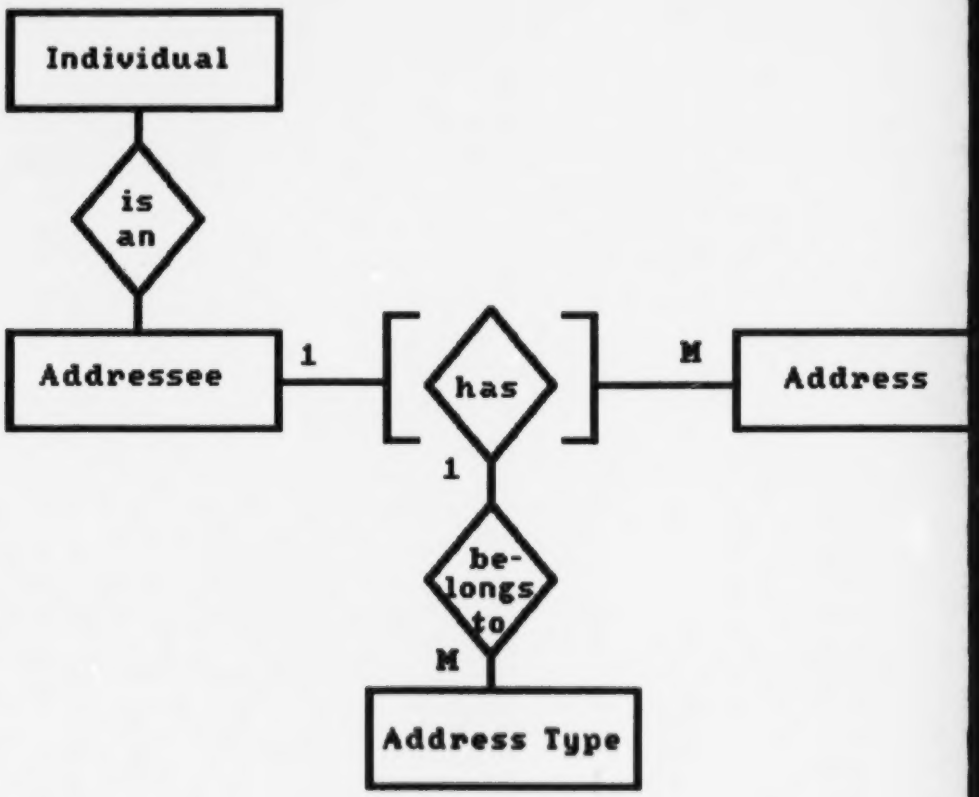
CONCLUSION

We started this presentation by describing the environment that started this development in motion. Then we described the steps we followed in implementing a solution; the importance of user participation, the issues that we faced during its design, and the richness of the features that resulted because of this effort. Let's visit with a few of the players to see how they are managing today.

Here is Libby looking at a copy of the 1989 Phone Book produced from the new Directory system. She says this is the earliest delivery of the Phone Book to the LSU community in years. Sandra is delighted with her responsive online Directory Assistance service. She is now able to make many of the phone and address changes directly and has retired her "master copy" of the Phone Book. And Ellen, now divorced, can finally rest assured that changing names is no longer as difficult as changing husbands.

Applications development starts with one user needing an automated solution for her/his business functions. Before you know it, you find yourself surrounded by lots of one-office solutions. What do you do? Face the issue. Integrate your systems, but in the process, don't forget to integrate your users too.

Figure 1. Directory Entity Relationship Model



Out of the Blue and into the Black: A Case Study of MIDAS

Timothy J. McGovern
Senior Project Manager
Network Services
tjm@mit.edu

MIT Information Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts
November, 1989

Abstract

From time to time, promising applications of information technologies bubble up to the surface. Ideas for new systems can be the result of some new technology becoming commercially available, or they can be born of previous efforts which didn't quite work. Sometimes these systems have no specific (read: traditional) owner or sponsor, that is, they are generic, community-wide information systems. At MIT, we believe that the IS organization must seize opportunities as they arise. We must carefully weed through all of the various possibilities and move swiftly to develop the most promising. The history of MIDAS, MIT's Information Distribution and Access System, is the story of just such a system. This case study examines the origins of MIDAS in previous failed efforts, its development history from a concept ("the blue") and how it was ultimately transferred to a production service ("the black").

I. Introduction

MIDAS stands for MIT's Information Distribution and Access System. It is an electronic file service available on a public (i.e., ubiquitous access) timesharing machine operated by Information Systems. MIDAS can be used by any two people or organizations to securely exchange data. Access to the service software itself is *not* restricted, but access to the data placed there is restricted.

For the initial period, MIDAS has focused on solving the problem of moving data from central administrative departments (custodians or data suppliers) to Administrative Officers (AOs) in academic department and research laboratories and centers (data consumers).

For a central department, MIDAS provides a single interface for data distribution for anyone in the community, without needing to bother with individual idiosyncrasies of the receiving user. Examples of custodians are Comptroller's Accounting, Payroll or Purchasing.

For a departmental administrator, MIDAS provides a single interface for receiving data from any of the many sources of data that they would normally need to work with individually.

"Although the present implementation of MIDAS is not a long term solution, it is a pathway for the transfer of a fair amount of administrative information, quite adequate until:

1. network authentication is available on the administrative mainframes, and/or
2. central systems invest in overhauling their data models to facilitate direct-inquiry access.[1]"

MIDAS has the potential to, but has not yet, changed the way MIT conducts its business with respect to how data is moved between the central administration and academic or research units. In the non-coercive climate found at MIT, change often comes slowly. In the end, MIDAS can be judged to be successful if fewer custom systems are developed to distribute transaction data.

"Without a business plan that everyone owns, bottom up projects [MIDAS is largely a bottom up project] can only go so far toward changing how we do business. Working bottom up risks:

a. picking the wrong "bottom" items on which to work
b. going off on tangents on one bottom item that worsens another.[2]"

1email communication with M. McMillan, October 18, 1988

2Ibid

II. Background

The Information Technology Organization

The responsibility for the computing infrastructure resides within the Information Systems organization. A concrete definition of what constitutes that infrastructure is underway, but includes at the very least, the telephone system, the campus data network, the supercomputer facility, a set of shared mainframes, both IBM and Digital Equipment Corporation machines which are used for a variety of administrative applications, and a set of general-purpose applications.

The Vice President for Information Systems is the Chief Information Officer (CIO) at MIT. The responsibilities of that role are defined in a document entitled *Administrative Computing Principles for MIT*[3]. This document also defines the basic criteria that the Senior Vice President (and a Steering Committee) can apply when evaluating proposals for systems development projects. These principles (and their implementation) are expected to evolve as they are applied to a variety of projects. Two of these principles are important to understand in the MIDAS context, the infrastructure responsibility and the call for innovation.

The *Principles* require that all organizations at MIT seek out and apply innovative information technology solutions whenever appropriate. Until October, 1989, one organization of the Information Systems group, Architecture & Strategic Technology (AST), was specifically charged to provide leadership in this regard. This group provided staff support to the CIO for innovative projects that did not have other sponsors, or were purely conceptual. This group was responsible for the definition and development of the MIDAS service during the period, April 1988 to January 1989. With the dissolution of AST, the responsibility for innovation has once again been distributed to each of the line organizations in IS.

Previous Efforts

In the years that preceded MIDAS, various efforts both at MIT and elsewhere provided experience from which to draw lessons about what to do, and not to do. Typically, these projects were tactical exercises, dreamed up by one or two people, often done with only minimal senior level support, usually done on a shoestring budget, and in technologies that were at the time not completely well-understood. These factors produced projects that didn't fare all that well in the end. These projects sooner or later encountered either technological difficulties, or political difficulties, or both. Here are a couple of the projects that the MIDAS project team learned from.

Technological difficulties

Several years before MIDAS appeared, there was a project called the Statement Display System (SDS), and a companion project, the MIT Accounting, Purchasing, Property System (MAPPS). The former was a mainframe based database system containing huge amounts of financial information, mirroring the existing batch financial systems. MAPPS was designed to deliver a read-only version of the information with conventional micro-mainframe file transfer technology (usually at very low speeds) to an IBM PC. At that time, that information was available only on printed accounting statements.

---

[3]James D. Bruce, Administrative Computing Principles for MIT, August 1, 1989

At the time of this project, many of the larger departments and labs at MIT already had invested in microcomputers of one form or another, and had begun to develop effective, if crude, office automation systems for managing personnel, financial and other resources under their responsibility. SDS/MAPPS strode into this environment, and was one of the earliest attempts to marry mainframe and microcomputer technologies. Unfortunately, it did not meet all of the needs of its principal constituency, the administrative officers (AO) in academic departments and research laboratories and centers, lacking among other things, the ability for an AO to run adhoc queries against the downloaded data. But the thing that really killed the SDS/MAPPS project was its poor performance. The value of getting some information, even if slowly, was not great enough to enough people to get them to change to a new way of "keeping their books."

Lesson learned: System performance is a significant, necessary but not always sufficient condition for success. Build systems in well-understood technologies, sacrificing some function, to make sure that the resulting system can be tuned sufficiently to keep the consumers happy.

Political difficulties: getting them to say yes & really mean it

In the summer and fall of 1985, Information Systems conducted a study that ultimately led to the publication of *A Proposed Administrative Information Systems Strategic Plan*[4] in the spring of 1986. One of the themes of the *Plan* was the need for widespread distribution of data from the central data stewards to the academic units. In order to study that problem, a Pilot Program (the Accessible Employee Database (AED) project) was instituted to distribute Personnel data to the departments. Prior to this project, an administrator relied on two techniques to manage personnel data (or any other for that matter) in local offices, SneakerNet and rekeying of data from printed reports. Although the AED made some progress toward eliminating use of these techniques, some eighteen months later, the project finally ground to a halt, for several reasons, among them:

1. the project went on too long, and without a well understood project plan, expectations that had not been properly managed in the early stages gave rise to poor morale and suspicion. The AED project in fact exposed many problems, both technical and organizational, which threatened the project's original scope and focus.

2. the administrative officers who were participating in the pilot project had long been skeptical of the central administration's desire to "help them" by distributing data, and as time seemed to slip away, they felt vindicated and either dropped out or lost interest.

Lesson learned: Keep projects short, work to a very tight project plan, and above all DELIVER!

3. the project had produced too little, and too late, to maintain the interest and energy of the departmental representatives, and the IS representatives. Most of the people assigned to the project had volunteered 25% of their time, for six months, and were not generally being rewarded for their altruism in their home departments. The IS staff involved were still responsible for all of their normal work during the time that the pilot project was going on.

Lesson learned: To the maximum degree possible, don't use volunteer help to do something REALLY IMPORTANT! When push comes to shove, the operational stuff has to get done first.

---

[4] J. Bruce, I. Colbert, C. d'Oliveira, A Proposed Administrative Information Systems Strategic Plan, January 21, 1986

4. the Personnel Office was unable to maintain the level of commitment to the project's vision during a critical personnel change. One of the *Plan's* principal authors was an officer in the Personnel office, and could make commitments to distribute the Personnel data. Within six months of the start of the project, that person had moved on, and with him, the Personnel Office's only understanding of the vision of the *Plan*. Although the pilot project continued for another year or so, there was never the same degree of commitment.

Lesson learned: Alliances with any one organization can be perilous if the champion in that organization has not transferred the "ownership" of the concept to the larger organization in which he/she sits.

As early as November 2, 1987, there was a MIDAS concept on the table. Given the demise of the Pilot Project effort, there were suggestions that any follow-on effort in the area of personnel information should be based on a generalized piece of software (as MIDAS ultimately proved to be) rather than be a custom office-specific data delivery system.

Fortunately, most of the lessons above were heeded during the MIDAS project. It helped that all of the developers of MIDAS had had direct experience with one or more of the above systems. So how did we proceed to a successful system?

III. "The Phoenix rises from the ashes"

The origins of the concept

After a second round of negotiations with the Personnel Office that tried to keep the Accessible Employee Database project afloat had failed, the project was abruptly suspended in March of 1988. As we were in the process of taking stock of what we had that could be salvaged from the Personnel project, and as we were looking around for new and interesting stuff to work on, fate intervened. There were a number of efforts going on around the now suspended project. At about the same time, the Financial Operations group had received a request for information from an influential member of the Strategic Plan's Advisory Council. Confluence also came in another form. Yet another group of AOs had been discussing ways that they could, working as a grassroots organization, build a personal computer application to manage the books of their (mostly research oriented) organizations. They petitioned for, and received, support for the idea from the office of the Vice President for Research. Thus there was a growing amount of pressure to begin to distribute important data. The Comptroller's effort had already done some preliminary studies and found that the SDS/MAPPS solution described above was not the answer, and they were getting ready to start to develop "son of SDS/MAPPS," another custom data distribution system.

The Vice President for Information Systems gave the go-ahead for a "proof of concept" project for an infrastructure service that would make the efforts of the groups mentioned above more efficient. This type of project solves most of the problems of the projects described above, as it limits the scope and duration of a given project, and use the "proof" as the mechanism for recommiting resources. With this charge, we approached the Comptroller with the idea of doing a joint development project. We proposed to develop and deploy a general-purpose data distributor system drawn from the technology developed in the AED. This system could then become part of the infrastructure. This would free the Comptroller's staff to develop any specific financial application software, on the mainframe or on microcomputers, that might be needed. As it has turned out, there are three application systems now in various stages of completion. The first is a fairly extensive mainframe-based system that permits AOs to examine, summarize and produce reports on their data in a variety of different ways. The data used is first retrieved from the MIDAS server, but left in a user's mainframe directory) rather than downloaded. The other two systems are personal computer based systems, one for the IBM PC family built using the Rbase product, the other for the Apple Macintosh family built using the 4th Dimension product.

Within three weeks of the initial idea, we had established the MIDAS Project organized upon the following principles:

- **Owner:** the Vice President for Information systems is the nominal "owner[5]" of MIDAS, in his role of CIO.
- **Custodians:** no single custodian manages the MIDAS facility. Any recognized data steward can make use of the system, for the price of the storage. To the maximum degree possible, a custodian should be self-sufficient. Although custodians alone can decide what they will distribute, clearly they aim to be responsive to the stated needs of the AOs.
- **Subscribers:** no single data subscriber's needs are more important than any other. The system aims to make data available to all platforms via use of commonly available protocols.

---

[5]Our chosen development methodology, Productivity Plus, from DMR Group, Inc. includes a very precise meaning for "owner." A system owner is the person or persons who pays for the system, who therefore is entitled to making the go/no-go decisions.

- **Development:** the system would initially be built with existing tools and software products, and include an API toolkit for use by custodian-provided applications.
- **Operations:** the system should not require additional operations, production control or other administrative personnel. That is, it should not require intensive administrator.

What it is & what problem it is trying to solve

MIDAS is a "smart" file server that provides a single, consistent interface to data providers, and a single, consistent interface to data subscribers. Its "smarts" derive from the fact that unlike some other data sharing mechanisms and file transfer schemes, MIDAS "knows" what files it currently has stored, and the characteristics of each file, such as the name and type of each data field. Its most important features are:

- providers are insulated from the needs of any single subscribers' workstation type and software
- subscribers are insulated from the existing data formats of any single providers' mainframe database system or application
- subscribers can request only certain fields of a given file, and in a particular sequence to aid in loading into pre-existing applications (both on microcomputers and minicomputers)
- providers are responsible for maintaining the definitions of their files, and a list of the recipients of those files
- there are hooks for custodian developed application software
- access to the system for all of the above is ubiquitous.

The "little green light"

The development team was given a challenge--the entire development cycle, from concept exploration, through analysis and design, programming, testing and documenting, would consume no more than 2 months, approximately 11 weeks from start to finish. This is what we have since referred to as the "little green light." Needless to say, we were going to have to be very judicious in our use of time. Unlike previous efforts, this one would not start out with an unlimited time horizon. This ultimately had the beneficial effect of limiting the resources that were poured into this project, both in the beginning and over the long haul.

The guiding principle for the project was this: fulfill a simple need with a simple product. The first version could be very crude, but it must be respectable. On the pragmatic side, we needed to get financial data to users more easily than it was then possible. We decided that quick turnaround prototypes would be necessary. As a guiding principle, we believed that throwing away a couple of prototypes was not tragic. Throwing away the user interface would have been tragic. To do it a "piece at a time," doing whatever proved to be usable and possible, was the right next step.

Seeking "proof of concept"

The MIDAS Project Team was given the go-ahead with the proviso that we could proceed until "proof of concept" or until it was clear that this product wasn't going to work any better than previous ones. Throughout the development period, we tried to define carefully just what this "proof of concept" would mean, and how we would recognize it when we saw it. In the end, "proof of concept" is very much in the eye of the beholder, but the better the beholder, the better the proof.

We allocated the time we had in the following way:

- Week 1: Sell the idea to potential data stewards
  Explore the concept
- Week 2: Complete concept exploration
  Do analysis of system in its entirety!
  Start technical design of system
  Start prototyping in several key technologies
- Week 3: Complete technical design and hold design review
  while prototyping continues
- Week 4-7: Build, test, and deploy server code
- Week 6-8: Build, test, and deploy client code
- Week 7-10: Complete QA and documentation
- Week 11: Deploy system officially; receive initial data feed from pilot custodian

Definition of the system

One of the difficulties that immediately faced the development team was how to read consensus on the features of a system when there wasn't really a traditional owner. It was fortunate that the development team had participated in the previous attempts to distribute data at MIT, and understood some of the obstacles that had been encountered. For that reason, we decided to take two different tacks to the ultimate system. The first tack would be to use conventional systems analysis techniques, with the development team playing all of the roles of a typical systems project, both the customer and the analyst. At the end of a week of sometimes excruciatingly long interview/design sessions, we emerged with a system design that remarkably enough remains an effective design document even now, some 18 months later. This design was presented, reviewed and approved at a formal design review meeting at the end of Week 3. The second tack was rapid prototyping of the system's client and server functions. In order to do that effectively, we needed to choose a development environment that was familiar to the developers, rich enough in tools to permit such rapid prototyping, and that would ultimately be robust enough to act as a platform for the emerging service. We found that all three of these conditions were met by the only ubiquitously accessible mainframe on the campus for general purpose time-sharing access, an IBM 3083 running VM/CMS. The system was prototyped and ultimately deployed almost entirely in the VM/SP System Interpreter language REXX, with a few routines written in PL/I for the sake of efficiency.

Development of the system

The use of proven technologies from the outset probably did more than anything else to ensure completion within budget and on time. In this case, the innovation was largely in terms of the organizational impacts of the system.

But even in a project that is using off-the-shelf technology, you need to have some way to capture and manage the results of the many brainstorming sessions that are inevitable in a development group. A simple "Futures" list, maintained electronically and shared frequently with the development team, worked very nicely. The MIDAS administrator still uses this list for planning purposes.

Our original goal was to have a single group provide sole support for all MIDAS-related activities--design, development, maintenance, training, documentation, support, marketing, planning. This would mean that there were far fewer communications paths to handle on a day-to-day basis. In the course of the prototype phase, we found that we needed to bring in a few other groups to resolve problems or provide additional support where the expertise did not exist inside AST. The fact that we had already briefed the Vice President and his direct reports (directors of the various

Information Systems departments) meant that we weren't delivering any surprises to the other (eventual) support organizations.

Given the tight schedule, we decided to limit the group that could provide functional input to just a few users and one eager custodian for the pilot stages.

Providing support to users while still developing the system could have swamped the development team. We decided to groom the initial set of four users so that they could then provide some limited forms of support to a larger (15-20) group of users. We put in place a "buddy system" between the "expert" AOs and the large group. Administrators are already more likely to call another AO for help with some type of administrative procedure questions than they would call anyone in the central administration.

Deployment of the system

The responsibility for choosing test subscribers was assigned during the "proof of concept" phase to the initial custodian. One of the issues that was batted around quite a lot was whether this group should include experts, that is, experienced computer users, or a group that was more representative of the community at large. Support for the former was predicated on the idea that what was *not* needed during the pilot was a lot of problems of supporting novice users, that the proof of concept was not on how well the system could be deployed, but rather on whether data distribution could work at all. The "representative" argument was that proving the data distribution could work for savvy computer users was no proof at all, that the real pudding that needed testing was the relatively inexperienced personal computer user. There was passion on both sides, but in the end, we decided to go with mostly experienced users, and to use the buddy system for less savvy users, to keep the demand for support from the central organizations fairly low during the pilot stage.

Reaching "proof of concept," or declaring "success"

By the end of calendar 1988, we had generally proven that the concept was viable, and our project was judged a SUCCESS. In the course of the project, we discovered a number of things. In summary, we found that:

- it is technically feasible to construct in a *short* time a "*smart* data server to accept and redistribute MIT's administrative data using off-the-shelf components
- administrators will need to (and will if allowed to) connect to data distribution services in a variety of communication modes, and at speeds far exceeding 1200 baud, including high-speed access over the MITnet
- custodians do not have to be responsible for maintaining "user profiles" that tell them what kind of machine, software, etc. exists on each user's desk
- custodians do not need to generate "special purpose" files where those files are merely a subset of the data fields found in an existing "master" file
- custodian effort can be kept relatively low (for the reasons in the previous two findings)
- users found the initial threshold low enough to warrant their continuing in the project
- launching custodian applications from within the data distribution system has both rewards and hazards on a number of dimensions
- data distribution is a necessary, but not sufficient, condition for the integral and effective use of information in departmental offices. The lack of workstation-based applications to process these data was a major limiting factor to the overall data distribution service.

Developing a service plan

Information Systems has formalized the process of bringing a new (not anticipated by the 5 Year Plan) service to the campus. The primary instrument in this process is a Service Plan. Among other things, the Service Plan describes the benefits of the service, the service's objectives, provides an assessment of the demand and need for the service, risks, alternatives, estimates resource requirements, and suggests possible metrics and evaluation criteria. The Service Plan is typically presented by a Service Champion, and reviewed by the IS management team.

Finding the most appropriate "service champion," and working with them on the Service Plan were therefore our next major steps. The MIDAS team began by developing a number of scenarios for the transfer from "proof of concept" to production. This included placement of the marketing, custodian and consumer support, and system maintenance and enhancement functions. We reached agreement with the Production Services manager that his group should be the Service Champion. The scenarios also called for a variety of different levels in the expansion of the user and custodian base in conjunction with the other support areas. These scenarios were presented to the IS management team in December of 1988, at which time the decision was made to pursue a low-growth option for the initial 12 months (roughly calendar 1989), calling for 25 new users, and 5 more custodians.

A major prerequisite for the wider deployment of the system involved the establishment of "computing budgets" for the academic departments. Although this might not appear to be a major problem to some, there was little history of the use of central computing resources by these departments. The situation was further complicated by the overhead structures of these departments versus that of the central departments. In the end, it was decided that a portion of the central computer that hosts the MIDAS server, and in turn provides user access, would be allocated to academic departments and research labs & centers. This took the form of a MIDAS Grant Program, where each participating AO applied for a grant of monies to use the MIDAS (and related) systems. The MIDAS Grant Program is administered by the Production Services manager. The individual grants are monitored by the Administrative Officers in each participating unit.

Transferring technology

The Service Plan called for the MIDAS development team to train the new team that would be responsible for administering,and maintaining the system. This was done using a series of seminars and workshops during January and February 1989. During this same time, day to day responsibilities were being transferred to the new administrators in the Production Services group. During January, the original team constituted the first line of support with Production Services "watching" over our shoulders. During February, those roles were reversed. On March 1, 1989, the original development team was no longer involved in day to day support.

Finally, we transferred the responsibility for the documentation products to the Production Services group. These products were:

User's Tutorial	User's Quick Reference Guide
Technical Guide	Administrator's Guide
Custodian's Guide	General Information Guide

IV. Conclusions

From every failure, there are nuggets of gold that can be mined. There is no question but that the experiences in the several years prior to the MIDAS project provided the MIDAS development team with a number of insights and techniques that enhanced the project's chances for success.

Not everything that you touch will turn to gold. Changing the way large, bureaucratic organizations conduct business takes a long time. No single technology product is likely to instantaneously alter the business of MIT. MIDAS is just one component of many that together can make MIT a more effective and efficient enterprise, and a more enjoyable place to work. In particular, the lack of personal computer based applications, which would create a greater demand for downloaded data, has kept usage of MIDAS lower than anticipated levels.

Data definition is our Achilles heel. We needed to open the kimono about the problems that existed with inconsistent data definitions. Again and again, MIDAS (and previous efforts) risked running aground when the data was not of sufficient quality to be shared with departmental units. The fear of sharing substandard data was that it would put new and intolerable demands on the support resources of the central offices.

Marketing is key. You can't assume that in the hustle and bustle of everyday life, that new ways of doing business (regardless of how good they are) will be adopted instantly. You must continually market the usefulness of the system to both existing and potential, subscribers and custodians. In particular, it appears that it would have been a good idea to have provided for a consultant whose responsibility would have been to work closely with all of MIT's data custodians to make additional information available.

Source Data Capture. Electronic capture of administrative data at the desktop and delivery to the appropriate custodial organization may ultimately be what's required to change the way business is done.

Be lucky!

A Voice Bulletin Board System for Career Placement

by

Norman L. Thienel

University of Virginia

Charlottesville, Virginia

ABSTRACT

The Office of Career Planning and Placement at the University of Virginia has implemented a voice bulletin board system to electronically post verbal announcements for job openings. Registered users can access the computerized system by calling from any telephone equipped with touch-tone dialing - no other equipment is needed by the caller. The bulletin board system automatically answers calls, requests the caller to key in his identification number, and then recites job listings matching the career and geographical interests of the caller. The system runs unattended around the clock, except for an hour each weekday morning when database information is updated. Out-of-date jobs are automatically removed from the database, so all information relayed to the caller is current. The VBBS user benefits from remote access to valuable information, and the only cost accrued by the user is paying the phone company for any long-distance calls.

A Voice Bulletin Board System for Career Placement

Introduction

The Office of Career Planning and Placement (OCPP) at the University of Virginia in conjunction with the university's Administrative Computing Services have implemented a voice bulletin board system to electronically post announcements for job openings. Registered users can access the computerized system by calling from any telephone equipped with touch-tone dialing; no other equipment is needed by the caller. The bulletin board system automatically answers calls, requests the caller to key in his identification number, and then recites job listings contained in the database that match the interests and qualifications of the caller. The system runs unattended around the clock, except for about an hour each morning when the database files are updated.

Among the extensive computing facilities at UVa, is one very important one located in the OCPP office. Since 1986, OCPP has been using a 13-station PC network to support its efforts to assist students in planning and choosing career fields. Nonetheless, before the advent of the voice bulletin board system (VBBS), OCPP had to collect paper documents announcing job openings to make them available for review by interested job seekers who studied, lived, or worked in Charlottesville. The information, although potentially of interest to a large number of people, was not readily accessible to many who could have benefitted from it. To have access to this information, a prospective client of OCPP had to either visit the office in person or telephone a staff member to request information. In effect, the data was available only within the walls of Garrett Hall, the home of OCPP.

A manual filing system was used to organize and maintain the documents and to dispose of those that became out of date. This method was slow, cumbersome, and labor-intensive.

The new VBBS provides remedies for each of these two inefficiencies, offering significant advantages to both job hunters and to the OCPP office itself. The VBBS uses a conventional computerized database system to select and present jobs listings that are of specific interest to the caller. Out-of-town and even out-of-state alumni can access the database around-the-clock. Callers are able to "page" through the jobs by using any touch-tone as the signal to interrupt the computer's speech and to go on to the next option. All job positions that entered the system prior to the caller's last call are suppressed, unless the caller asks to hear all matches, regardless of when the job listings entered the system.

Benefits to the Caller

The VBBS clients are scattered about the country. When a job seeker calls, selections are made on the basis of the geographical preference indicated when he registered to use the VBBS. If the caller ever wishes to select on the basis of a different geographical region, he may utilize the on-line option to change his geographic preference, and then ask the computer to restart the matching process. At the opening voice menu, a caller can choose between hearing all matching jobs, or only those that have been added to the database since his last call. Out-of-date jobs are automatically removed from the database, so all information relayed to the caller is current. The VBBS user benefits from virtually instant access to information, and his only costs are a nominal registration fee and then paying the phone company for any long-distance calls.

Voice quality of pre-recorded speech played by the VBBS is indistinguishable from a live voice. The system is very simple for a caller to use: a voice menu is presented after each VBBS function completes. An introductory message is given to all first-time callers explaining the operation of the VBBS, and the system offers on-line help messages that can be requested.

When a job seeker calls the system, the VBBS software identifies the caller by his ID number (SSN), finds his data record, and makes note of the values for the three parameters on which it must find a match: major, geography, and career field. Then the job listings are searched, and when one is found that has attributes corresponding to those desired, the pre-recorded speech file representing name of the employer, the job title, and the short description is played to the caller. The caller can then have these items repeated, have a longer, detailed description played, go on to the next job match, or return to the previous menu of choices. And at any time during the playing of these speech files, the caller can interrupt by pressing a touch-tone, and the system will respond by playing the menu (which can also be cut short by the caller pressing a key) and asking for the caller's next menu choice.

Benefits to OCPP

This system provides a vehicle for OCPP to distribute valuable information to its client base in a timely and efficient manner. Using the features of the DBMS, OCPP can easily monitor usage of the system, knowing exactly who calls and how often, and which jobs are most frequently matched. The unattended system is available to callers for 23 hours a day, and at an operating cost that involves only the manpower required for entering the new data each day. (Out-of-date data is purged automatically.) The reputation of OCPP is enhanced because both employers and prospective employees recognize the effectiveness of the system, and usage by one group encourages greater usage by the other.

Development of the System

The design, programming, and implementation of the software for this system was performed by Administrative Computing Services at the University. The telephone management functions needed to answer the phone, accept input, and play voice files were provided at the dBASE programming level by dbSpeaker(tm)[1]. Because the VBBS system is essentially a dBASE database system, development time is short and the effort is very much like conventional dBASE programming. In fact, much of this system's program code was developed before the arrival of the Watson system by using keyboard input and screen output to simulate telephone input/output (I/O). Then, when the Watson voice sub-system did arrive, it took only a short period to convert terminal I/O to telephone I/O. However, interacting with a telephone caller does present several complicating issues because of concerns with time-outs for input, having touch-tone input interrupt speech, and handling situations where callers hang up in the midst of a session. dbSpeaker, though, does provide useful tools to integrate the features to conveniently handle these situations. For instance, when waiting to read touch-tone input, the VBBS dBASE program passes to dbSpeaker a parameter defining the time period to wait before "timing-out". If the caller does not respond in the allotted time period and there is, in fact, a time-out, the program tries again by repeating the question two more times if necessary. If there still is no response from the caller in the way of touch-tone input, the VBBS will electronically hang up the phone on its end. This is necessary because there is no direct test available to see if the caller is still on the line. Consideration was also given to not allowing the total length of the call to exceed some value, say 10 minutes. But it was decided that because most callers would be paying for long distance calls, it was not necessary to limit call duration. If, in the future, this feature becomes necessary, it will be a simple matter to alter

---

[1] dbSpeaker is one component of the Watson(tm) voice sub-system that is a product of Natural MicroSystems Corp. of Natick, Massachusetts. Other required components for this application are a Watson interface board that resides in the PC, and two other software products from Natural Microsystems.

the program to note the time the call began, and to periodically check to see if the caller's daily limit of on-line time has been exceeded.

Although the Watson sub-system can output synthesized voice messages for variable data (e.g., values for a job title, name and address of an employer), it was decided to verbally record all data that is relayed to callers instead of keying it and letting the Watson voice system synthesize it. This approach yields a couple of significant benefits. First, there is far higher quality for the system's output as every message heard by the caller is a reproduction of a human voice. (The image of the voice is stored digitally, and it can be reproduced an indefinite number of times without any distracting background noise.) By using recorded speech for the job listing information, the voice output is much clearer and the task of data input is made easier. The data input module was written to accept the spoken word for job data fields such as job title, name and address of employer, short and long descriptions of the job, and name of person to contact. These data items are then entered into the system during its one hour of off-line time each day by speaking them into a telephone dialed into a data entry program running on the VBBS PC. Speaking these descriptive items into the system takes much less time and effort than keying them. Voice data is converted by the Watson system into digital data which is stored as disk files. Data for geographical location, career area, and required degree still must be keyed; these value are used for the dBASE matching algorithm and therefore must be present in the employers' database file in a standard dBASE format. These values are never directly relayed to the caller, but instead are implicit in any job match data spoken to the caller. (It was also decided to key the name of the employer in addition to entering it verbally. This allows historical reporting to include data on company names.)

A standard IBM-compatible PC is an adequate platform for implementing a VBBS application. However, it should be noted that if job data is stored as digitized speech, the disk space requirements become substantial. The dbSpeaker software uses a default speech compression rate of 3 Kbytes per second when recording voice data to disk, so a five-second message, for

instance, will require 15 Kbytes of disk space. If each job has a job title, company name, short job description, long job description, and name and address/telephone number of a contact person, and then several hundred job descriptions stored on the system may require more than 50 Mbytes of disk storage. Considering this along with our expectation to upgrade the system in the future to accommodate multiple, concurrent incoming calls, an 80386-based PC equipped with a 60 Mbyte fixed disk was selected.

Registering the System Users

Students and alumni who wish to gain access to the VBBS are required to submit a completed application form to OCPP. On this form, the individual specifies his academic major, his career field interest, and geographical preference. To simplify the task of data entry at OCPP, each applicant writes on the form the actual code corresponding to his each of these three items (the entire set of codes is listed on the application form itself). Prospective employers, on the other hand, submit their hiring requirements on documents whose form and content vary widely. Therefore, keying this data often involves some decision making for the career field area or field of study. If the employer does not specify any majors that are required for the job, 'any' is entered and this will match with all student majors.

Management Reports

Because the system is essentially an automated dBASE database, it was a simple matter to program the usual report features that tell OCPP management about the usefulness of the system, and the characteristics of the clients who utilize it the most. Daily reports contain

data for percent of time the system was in use, number of calls received, number of matching jobs found, number of students registered, number of students deleted (when the three-month registration period expires), and number of jobs currently listed. Other reports that are useful over longer intervals are number of calls received during the period; number of different callers using the system; and total number of positions that have been advertised. Reporting options can also list which career fields, majors, geographical areas have yielded the most job positions, and which employers have listed the most jobs. Another report will produce mailing labels for all employers who have used the system (provided employer address information has been keyed).

Future Plans

As demand for this system increases, a multi-tasking system will be installed along with a second phone line into the system. If the caller load begins to tie-up the two-line system, then a network will be installed and the current machine will be converted to the file server.

Another potential new feature will be the implementation of a voice mailbox so that callers can leave voice messages concerning problems or suggestions for OCPP to review the next business day.

A Counseling Reservation System in a Local Area Network Environment

Bruce L. Rose

Cuyahoga Community College

Cleveland

Ohio

Abstract

This paper will discuss the development and implementation of the Counseling Reservation System developed at Cuyahoga Community College. In addition, the paper will discuss future plans for automation and Information Systems in the counseling and student services area - specifically, the expansion of the Metropolitan LAN and applications to CCC's Western and Eastern campuses. Finally, this paper will analyze some of the inherent problems found when introducing networked departmental computing into the College's offices as a case study which may benefit others following this same path.

I. Introduction

Cuyahoga Community College is a large two year institution which serves the greater Cleveland area in northeast Ohio. The College is composed of three distinct campuses and a district administrative office. The three campuses - Metropolitan, Eastern, and Western - vary in size, and composition of their student body but all three remain subordinate to the one "College" both organizationally and operationally.

CCC has aggressively pursued a leadership role in both administrative and academic information systems (IS) and computing, thus there are numerous IS initiatives which contribute to the College's strategic mission of academic quality, access, and success. One of these initiatives is the introduction of automation and networking to the counseling and advising areas of the College.

Beginning in 1986, a "pilot" local area network (LAN) and telecommunication capability was installed in the Metro campus counseling area in order to evaluate the effectiveness of providing both local resource sharing and host connectivity to the counseling staff (about 18 counselors.) The results were impressive in that most counselors used the electronic mail and word processing capabilities of the LAN in their daily work, and there was a general clamor for additional training, software, and resources.

In particular, the computing consultant (the author) and the director of the counseling area agreed that the key enhancement need for the LAN was some type of scheduling or appointment capability which would be accessible by all of the counseling and clerical LAN users.

A simple scheduling system was in use which provided batch reports of counselors' daily schedules as well as student record request lists. This Scheduling System originated from a timesharing application on the Honeywell/Bull mainframe computer, and was migrated down to the PC using dBase III's development language and database format. This simple Scheduling System would form the basis for the Counseling Reservation System which was jointly designed by the counseling director and the author.

The resulting application, the Counseling Reservation System, is operational at the Metro campus and will soon be migrated to the other two campuses. The system itself, has many merits and some weaknesses and is well liked and exploited by the users.

The development and implementation of this application will be explained in the following pages and represents the primary theme of

this paper. A secondary theme of this paper is to discuss the Reservation System development and its importance to the overall IS strategy in the Student Services and support area of the College.

There are some interesting characteristics of this system's development, which are also discussed. For example:

1] The scope of the system was technically and functionally conservative and was delivered rapidly to the user

2] The system was a literacy tool for both user and developer, and would be used to guide larger and more ambitious development efforts

3] The resource requirements were minimal and could be absorbed without committing or delaying other formal development projects

Section II is a discussion of the system development; section III discusses the implementation of the Reservation System, and finally section IV concludes by summarizing what was learned in this development and how others may benefit from this experience in small systems development.

II. Development of the Reservation System

The Counseling Reservation System grew from a simple scheduling system which was first developed on the Honeywell/Bull mainframe computer used at the College. This system was later migrated to an IBM PC application using dBase III's file format and application development language (ADL).

The scheduling system simply accepted appointment transactions keyed into a specific format, and later collated these into daily schedules and other reports which were used for requesting files from the records office.

The counseling director came to the computer center with the initial concept of a real time scheduling or appointment system which would eliminate the need for a manual schedule book. He had the following parameters to work with:

- About 18 full time and a few part time counselors; his staff generally worked from 8:30 to 5:00, but there were appointments taken in the evening from 5:00pm until 8:00pm and a small staff on Saturdays.

- Each appointment was 30 minutes in duration and was handled one on one with a counselor and student.

- There were numerous other "types" of time commitments for which the counselor would be assigned including walk-in counseling during registration periods and student orientations. In addition, the counselors enjoyed faculty status and might be teaching a class or have other academic commitments.

- For scheduled appointments, the director closed out the scheduling about 3 or 4 working days ahead of time. That is, a student could not normally make a scheduled appointment with less than three days lead time. This allowed the office schedules to be collated and distributed, and allowed the records office to deliver the students file folders to the counseling office.

In terms of design, we agreed that the technical problem of automating a schedule matrix would be solved first. The author completed the file design and a simple scheduling program or algorithm which would accept a time range stated in normal clock hours (e.g. 8:30am or 10:00am) and "map" or translate this time range into the scheduling matrix.

The scheduling matrix or table looked something like this when completed:

Date	Counselor	Time Slot #1	Time Slot #2...Time Slot #<n>
04/11/89	JM	CC	CC
04/12/89	JM	XX	
04/14/89	AM		CC

The time slots represented 30 minute time commitments, while the two-position code (e.g. CC, XX) represented the "type" of commitment; a blank, of course, meant the time slot was open or available.

This worksheet style matrix had the advantage of being very straight forward and lent itself very well to display and visual inspection. The intent of usage was clear at this point, and would follow the following events:

- The student would approach the front desk or call the counseling office to make an appointment to see a counselor.

- The operator or clerk would glean the intent of the appointment

(type), and if the student had preferences for the counselor (Who?), the time or date (When?).

- Armed with these preferences the clerk would query the schedule matrix and visually determine the best fit for the student, and schedule the appointment.

The key to this "point-of-sale" process was providing the strongest possible visual representation of the schedule matrix on the limited size and resolution of the Personal Computer screen or monitor. We determined three separate queries or visual displays would be necessary:

- A query by date. This displayed the schedule matrix and time slots for all counselors for a given date. The table could be "scrolled" or moved forward and backward in time using the PC's cursor control.

- A query by counselor. For a student who wanted to see a particular counselor, this display would show all available dates for one counselor.

- A query by BOTH date and counselor. For a student who had both a date and counselor of preference, this query would zoom in on the specifics of an individual daily schedule. This, by the way, was the same query which produced the daily schedules for the counselors.

The terminology, Reservation System, is best represented in this part of the application. The director and author envisioned an airline reservation clerk interacting with a computer terminal display to accomplish reservations. Speed, accuracy, and customer satisfaction would all be dependent on how efficiently and effectively the computer would visually bring forth the needed schedule information, and how the clerk would navigate to the desired "best-fit" between customer need and availability.

This was the most difficult part of the design in that beating or equaling the manual process (i.e. visually inspecting and updating the manual schedule book) was a challenge. If we could satisfy ourselves that we had at least equaled the manual process, we knew that the other benefits of the system would warrant a full implementation.

The system, was developed along these lines. A computer center analyst implemented the author's time scheduling "engine" and completed the application in about 3 months of part time involvement. The system was tested stand alone at the counseling front desk, and the manual scheduling book continued in use as the authority information. The preliminary results concluded that:

- There was a rather high rate of data entry errors. In particular, an incorrectly coded student number (social security number) would introduce duplicate and erroneous records into the system.

- The visual representation of the schedule matrix was still inadequate for effective scheduling.

- The scheduling time engine worked as advertised, and the resulting time savings were very good.

The most critical enhancement of our second testing phase was to add a significant database overhead to the system, namely the student authority file.

We agreed that we would compose, extract, and maintain a file of all registered students at the local campus to be used in the Reservation System. The source of this file would be the mainframe based Integrated Student Information System (ISIS). This addition would add significant complexity and overhead to the system, but would solve our data integrity problems and would create an interesting tracking dataset which could be queried statistically in the future. The addition of this component of the system had the following characteristics:

- This would be an operational database. The information would be extracted and transferred to the PC or LAN, as opposed to a realtime interaction with ISIS.

- The authority student information would be linked to the local counseling information such as number of appointments, counselor last seen, and cancellation or no-shows.

- The file would be cycled on a quarterly basis. Students who had an appointment history would be retained, and the "age" of the student's experience with the counseling office would also be tracked or known.

The addition of this file, meant that the clerk could assume the properly registered student would be "known" to the system. This saved keystrokes and added greatly to the data integrity of the student appointment file. The student could be told of what the "system" recorded as their current demographic and address information; thus, changes or inaccuracies would be quickly reported back to the records office.

This authority file was implemented and to date we have one and one-half years of tracking information in the file. The preparation and transfer or "downloading" of the authority file is somewhat formidable (about 5-10 thousand records of 200 characters each), but once the file is operationalized the capacity of the local software and hardware is not really challenged.

III Implementation of the Reservation System

The formal implementation of the Reservation System followed the second testing period. This testing period continued to use the manual schedule book as a backup, but more and more reliance was placed on the automated Reservation System. The results were very good and the system was considered operational at the Metropolitan campus.

Chronologically, we set forth the following milestones:

A stand-alone Reservation System	Jan 88
Access by all counselors	Jan 89
Migration to other campuses	Jan 90
Multiple updaters on the system	March 90

In small systems development, it is typically a surprisingly large leap to move from a stand-alone or single user application to one which is networked or provides for multiple access. The technical tools provided on the smaller systems are not geared for multi-user systems, and in general the data becomes much more vulnerable when "opening" up the system to multiple access.

As mentioned, the counseling area was equipped with an Ethernet local area network which originally ran 3Com's EtherSeries network software (this is now Novell Netware). The physical and logical network components were in place to allow access from the counseling offices, but there remained the technical problem of modifying the application to support multiple or concurrent usage.

We proceeded cautiously in this implementation. We clearly separated the functional milestone of adding additional "queriers" to the system and the milestone of adding "updaters" or the ability for two users to concurrently change the data. We understood that from a needs perspective, the counselors wanted the following from our system:

- o inspect their schedules from their own PC workstations
- o inspect a student's record or history with the office
- o inspect their peers' schedules and commitments

for which all three of these needs precluded any changes to the database; thus, they needed only to read or query the files as opposed to updating or changing the files.

The development environment, namely dBase ADL and a dBase compiler called Clipper (Nantucket Software), allows for multiple user capability. However, the lack of built-in or supplied data integrity and access control features places a considerable burden on the local programming and application design work.

We have, quite simply, moved slowly on this front because of a concern of losing reliability and simplicity in the system. It would be disastrous, at this point, to introduce multiuser database problems to a system which has such an excellent track record of reliability.

Instead, we have completed a query "module" or separate program which is a subset of the full Reservation System. This module insures no updating or changing of the system's datasets can be accomplished. This module is accessed by multiple counselors from their workstations and gives each counselor the ability to inspect or read all of the Reservation System's scheduling and student information.

We will migrate this current form of the application to the other campuses during this year. Our final milestone, is the completion of multiple user appointment updating of the system. This would be most helpful in times of high student activity and would allow the system administrator the ability to change or alter information while the Reservation System is in use. Additionally, it opens up enhancement questions such as counselors scheduling their own appointments and adding comments or tracking information to the student records file.

IV. Summary and Recommendations

Readers knowledgable on technical matters regarding PC's, LANs, and networked applications might question the degree to which the developers urge caution and a conservative approach in developing these types of applications. There are certainly products and development tools which promise such things as true multiuser database management and integrity controls in this environment. However, it has been our experience that cost, and a lack of prevalent technical expertise can mitigate their usefulness.

As a case study, the development of this Reservation System is likely very representative of efforts taking place across the country. These development efforts are quite entrepreneurial in nature and typically involve a key "user" (i.e. our counseling director) and a sympathetic consultant in the computer or MIS center. We readily endorse this model, and observe the following advantages:

- The vision and enthusiasm for the project is very focused which enhances the likelihood of success and decreases the likelihood the project will get sidetracked or continually redefined.

- The project is narrowly defined, technically conservative, and can be delivered quickly. In this model human and technical resources are always at a premium; this necessitates the project "team" utilize and stretch their available resources, and implement before the resources "dry-up."

But, we have the following caveats as well:

- Always search the market for an existing solution or one which can be customized to fit your needs. This can be a difficult "pill" for the designer and developer who are excited about applying their own solution.

- The project must fit into some overall strategy or context. The end result should add value to larger or more conspicuous development or IS initiatives, and should fit into these efforts.

At CCC, this Reservation System development has been an entrepreneurial enterprise. The design, implementation, and expansion of the system has taken place with minimal resource impact and has been largely a product of one key user's innovation and hard work. However, the system does have an excellent fit with larger development strategies.

The counseling and advising areas, in addition to their networking and telecommunications capabilities, will shortly benefit from the delivery of a Degree Audit system which is built around their PC workstations and micro to mainframe communications and interfacing. This Advising and Graduation Information System (AGIS) fits very well with the Reservation System. Each serves a separate purpose, but the access, look and feel, and training and orientation of its users go hand in hand.

In all of this technical detail it is important to remember that the College's technology motivation is not about computers and hardware and wires. Its about quality and effectiveness. The opportunity to work smarter in the College offices, and to improve the quality of instructional programs, and most importantly to enrich the lives of each and every student who walks through our doors. In its own small way, we feel this system contributes admirably towards this goal.

CASE TOOLS for the 90's: DELIVERANCE OR EXTRA BURDEN

Dr. Paul J. Plourde
Bentley College
Computer Information Systems Department
Waltham, Massachusetts

This paper examines the viability of CASE (COMPUTER AIDED SOFTWARE ENGINEERING) as a replacement for the design and development strategies that have been employed in the past: namely, manual design and programming of application systems.

The primary focus will be on developing evaluation questions and assessing whether or not industry reports support the notion that CASE tools boost productivity and reliability of systems that are developed using these tools.

Consideration will be given to the questions of the necessity for the integration of CASE tools with other tools such as data base management systems, 4GL's, as well as, other CASE tools.

During the early 1970s, computer centers and systems development departments were abuzz with a new technology in the form of data base management systems (DBMS). Systems analysts and programmers were charged with determining the capabilities of the commercially available DBMS' and ascertaining what, if anything, such systems could do for their organizations.

The results of these investigations are history and many organizations adopted the use of these systems to provide more timely and integrated information for their users. However, the impact on the systems development organization was not as smooth as this scenario might imply. In the first place, there is a difference between acquiring a tool and altering the methods (techniques) used to develop systems. In the case of DBMS, acquiring a DBMS package did not insure that it would be utilized for something other than a replacement for the current disk accessing technique whether it be random or some form of indexed sequential. It was quite another matter to have analysts think in terms of an integrated data base for the organization and to develop systems with this concept in mind rather than focusing on the single system approach to development.

As we enter a new decade, we are faced with a new buzzword (CASE) which promises to revolutionize systems development and have an impact that could be more profound than that which we experienced with data base management systems. Even with this promise, the potential problems of integrating the use of CASE tools in an information systems organization are as foreboding, if not more so, than was the case with incorporating a data base management system (and the associated data base philosophy) into the tool-set used by systems developers.

The purpose of this paper is explore this new technology, indicate how it evolved, as well as, describe its current status, and identify some evaluation criteria and problems of integrating this new technology into an existing systems development organization.

It is difficult to cite an exact definition for CASE (Computer Assisted Software Engineering) but there is a common thread to many of the definitions that have been advanced. There seems to be a consensus that CASE deals with automating (applying the computer to) the systems development tasks and that the goal is to increase the quality of the software produced and improve control over the process of developing systems. From this definition, it is conceivable that a broad variety of existing tools could fall under the rubric of CASE tools including:

1) analysis and design tools (front-end or upper case tools including graphics, data repositories and process definition systems,

2) programming support tools (back-end or lower case tools) including code generators, screen painters, optimizers, editors, as well as, reverse engineering tools, and

3) project management tools.

A more specific definition of the components of CASE is offered in the many reports issued by James Martin on this subject and he suggests that there are five components of CASE:

"1. front-end design and specification graphics support, which at least relieves the analyst of the manual chores related to drawing and redrawing diagrams as the design evolves;

2. design analysis, which at least tracks and reports basic design flaws such as design pieces that are not related to any other piece of the graphic presentation. Some of the rule-driven tools now emerging can also detect other inconsistencies;

3. code generation, providing automatic translation of the specifications developed by the earlier components into source or machine code;

4. a "repository," "encyclopedia" or "metadictionary," which holds comprehensive entity models or views of the structure of the organization using the CASE facilities; and

5. "absolutely essential to the effective use of other elements" is a PC or similar commonly-used processor, which in addition to being a familiar, non-threatening and easily accessible piece of equipment, is (under CASE) provided with "very good interfaces including windows and menus and color." (Leavitt, pp.50-51)

With this admittedly sketchy understanding of what a CASE tool can accomplish, let us consider how we arrived at this juncture. In what must be considered a rapid succession of events, we have proceeded from developing computerized solutions to business problems by generating machine language code, to symbolic code, to compiler languages, to fourth generation languages and to the present state of CASE which is a specification-driven language that may or may not be tied to an automated code generation facility.

The infiltration of CASE into systems organizations is reasonably well documented but studies of their effectiveness in producing quality systems and/or reducing the backlog in systems are in the early stages of being undertaken and the results have just begun to appear in the last two years, Since few integrated CASE tools such as TI's Integrated Engineering Facility are on the market, statistics on utilization of CASE tools and their various derivations can be misleading. The norm for a company seems to be to utilize either an upper or lower case tool in a stand-alone fashion or they use an upper-case tool from one company coupled with a lower case tool from another company.

Based on the number of users of CASE tools on mainframes, one would not necessarily assume that there is and will continue to be a tremendous growth curve in this industry. As Myers noted (reporting on a study conducted by Computer Intelligence Corp.), "only 2% of U.S. mainframes have one (a code generator) installed" (Myers p.49). However, such figures are quite deceiving since much of the current development in the use of CASE tools is with front-end tools or is PC based. Index Technology's Excelerator, which had 23% of the CASE market share in 1987, as reported by the Computer Systems News, reported in late 1988 that it had shipped the 10,000th copy.

In a compilation of eighty-five (85) CASE software products included in a March 1989 Computerworld article by Santosus (pp.77-86), thirty-five (35) were PC based and another twenty-one (21) were workstation based (SUN, APPOLLO, VAXSTATION etc.). Only thirty-two (32) products worked solely with mainframes to the inclusion of PC's and workstations. The fact that many of the PC and workstation products also worked on the mainframe or had mainframe interfaces suggests that as one moves towards an integration of upper-case tools and lower-case tools the mainframe is more likely to be involved.

WHY CASE?

It is a well documented fact that in most information systems organizations have backlogs of requests to develop new systems or maintain existing systems. Added to this is the geometric rise in the number of computers and the declining number of graduates from computer science and computer information systems programs. The result of these factors as noted by McFadden and Discenza is that

> "A crisis in software development plagues American business today..In total, the number of new applications ... in the backlog undoubtedly exceeds the number of all existing applications...and the (IS) organization cannot respond to the need for new systems." (McFadden and Discenza p.68)

The net result of more computers, fewer trained IS people, and a huge backlog of requests for new systems and changes is that we cannot hope to satisfy our organization's systems needs with the use of traditional tools. Some observers suggest that increased user involvement with systems development is the solution while others such as McFadden and Discenza argue that the solution lies with use of fourth generation languages. An increasing chorus can be heard in favor of increasing productivity through the use of CASE tools as a complement to user development and using 4GL's and other user-oriented tools.

It is not surprising that accurately measuring productivity is and will continue to be a major focus in determining whether there is a payback in using CASE tools. Barry Boehm, who developed what some consider to be a landmark survey of productivity tools in the 1970's, has done some tests and reports on the perceived payback of CASE that

> "when developers were given their own workstations and a set of tools that covered the entire life-cycle, not just programming, a productivity gain of 50% was found."
> (Knight p.56)

Similarly, Necco, Tsai and Holgeson, in a study conducted in 1989 to determine if industry has aggressively tried to implement this new tool, found that

> "only twenty-four percent (24) of the responding companies indicated that they were utilizing a CASE tool... and a majority (60%) acknowledged that the CASE tool significantly improved the quality of the product, but only 47% indicated a significant improvement in productivity."p.8

Necco and his colleagues further report that

> "An analysis of the factors that influenced these improvements reveals that more than half of the respondents noted a significant improvement in the communication between the analyst and the systems users. Sixty percent, however, indicated that the tools did not improve project control, and would not make future maintenance changes easier." pp.9-10.

In a study of 3,000 active users of front-end CASE tools in the U.S., the Barton Group found that

> "Users report that exceptionally strong and widespread gains are made in documentation...
>
> A very large group reports respectable but not extreme improvements in... the quality of systems design..

> Many users report strong improvements in ...the ability to meet business requirements...
>
> Responses indicate a widespread, mid-level improvement in communication, as the tools force standardized outputs and documentation sets.
>
> Most projects experience moderate improvement..in productivity..more time is needed to improve ..productivity
>
> Most people experience minor improvements in..project schedules. The learning curve is greater than expected, but this is offset by improved documentation and communication."
> (Merlyn and Boone p.66)

Regardless of the preliminary results on the success of CASE, supporters contend that benefits will be derived. Charles Martin in a vendor sponsored meeting of Excelerator users suggested that productivity improvements can come from four areas:

> "Methodology training and enforcement--
> The CASE tool helps train junior analysts in advanced techniques and enforces consistent usage throughout the organization.
>
> Support for systems analysis diagrams--
> Interactive graphic editors help analysts to develop the kinds of process, data base, and program structure diagrams which have proven to be the most effective way to communicate concepts behind the requirements and design.
>
> Single entry specification bookkeeping--
> Operating from an Information Resource Dictionary System, redundant specification documentation (viewing the specifications in different ways) can be prepared from non-redundant dictionary representations.
>
> Reminds and consistency checks--
> Complex inter-related bookkeeping chores are eased by CASE tool reminders of additional information needed to complete specifications and automated consistency checks." p.56.

CASE EVALUATION CRITERIA

Until we encountered the introduction of DBMS in the early 1970's, many of us were unfamiliar with software RFP's and such exercises were limited to the acquisition of hardware.

Since that time, software RFP's have become more common for all but the smallest computer installation. Many among us have drafted RFP's for application software, LAN software, screen painters, 4GL's and other software. Thus, it is not surprising that the first admonition is to study the company,its direction and the product direction. Just as there were many start-up companies in the DBMS field that didn't survive the maturing of the technology; such may be the case with vendors of CASE tools and some suggest that this phenomena is already repeating itself in the area of CASE tools. As Grochow of American Management Systems notes:

> "there were 50 code or application generator producers in 1982 when first became involved in the technology. Today,...of those 50, 10 are still around but there are 40 more. It's a volatile market." (Myers p.68)

A number of articles suggest evaluation criteria and checklists for identifying which CASE tool or set of CASE tools might best suit the needs of your organization. While most articles include a checklist of sorts, the articles by Aranow, Gibson, Santosus, Troy and Zucconi focus in particular on this issue.

Beyond vendor issues, some of the major questions are listed below with each of these questions having numerous sub-questions.

EVALUATION QUESTIONS

Does the product SUPPORT THE ENTIRE LIFE CYCLE and is this provided by a set of products as opposed to one product? If a number of products are used, how well are they integrated?

Is SUPPORT provided for the MAJOR DEVELOPMENT METHODOLOGIES such as Demarco, Gane and Sarson, Yourdon etc. ?

GRAPHICS CAPABILITY (DFD's, Flows, Action Diags, Models

Does it SUPPORT LOGICAL DATA BASE MODELLING?

Does it SUPPORT A CENTRAL REPOSITORY?

What PROTOTYPING CAPABILITY is provided? Is this an industry capability or is it product specific?

What PROJECT MANAGEMENT TOOLS are provided and supported? Are these industry compatible or unique?

Is this a MAINFRAME and/or PC BASED system? If both, what interfaces exist?

Is there provision for MULTIPLE USER ACCESS?
Does this include support for major LANS?

Is it INTEGRATED WITH MAINFRAME DBMS & DATA COMMUNICATIONS with LANGUAGES, DSS, 4GL'S & other CASE TOOLS

Is there CHECKING OF SPECS FOR COMPLETENESS & CONSISTENCY

Does an EXPERT SYSTEM EXIST TO CHECK QUALITY & ACCURACY OF DESIGN?

Does the system SUPPORT AN OPEN ARCHITECTURE PHILOSOPHY?

What are the HUMAN INTERFACES and TRAINING TOOLS?

POSSIBILITIES FOR SUCCESS

Having evaluated and selected a CASE toolset, what are the possibilities of successful use, are there any strategies that will help and what problems are likely to be encountered and can they be avoided? Merlyn and Boone suggest that "success is not an automatic outcome of CASE" (p.68) and they cite a list of suggestions compiled as a result of the Barton study (noted above) about what organizations should do to derive the maximum benefit from the use of CASE products.

The suggestions from the Barton Group study are as follows:

"Establish a means for measuring results that addresses both short- and log-term costs and benefits.

Keep expectations realistic. Look for short-term improvements in communication and the quality of deliverables, but do not expect major improvements... for at least three years.

Move slowly and carefully....organizational changes are required therefore move incrementally

Scout the territory. Companies that understand the methods first have a better chance of success.

Test extensively...conduct at least four pilot projects

Forgive test errors. Expect to make mistakes..

Allow for postpurchase expenses.

Splurge on training...and.. supply coaching.

Focus on use and support ..and.. encourage full use.

Address organizational issues.

Make improvement a strategic goal.

Involve the project manager." (Merlyn & Boone pp.68-69)

Burkhard also suggests that certain guidelines be followed to insure success and that "like all correct development cycles, the implementation plan must be committed to a structured methodology." (p.21) Danziger and Haynes echo these findings. The adoption of a new technology is often accompanied with problems and CASE implementation is not an exception. There aren't any standards and there are scores of vendors marketing all sorts of products under the aegis of being a CASE tool. As was the case with local area networks the major hardware manufacturers have stayed out of the fray until recently when IBM acquired a percentage of Index Technology while at the same time pushing forth its own AD/Cycle Information Model (see Hazzah Dec.89 for a discussion of AD/Cycle).

A significant problem which again parallels the data base experience is that CASE technology is ahead of the CASE techniques that people employ. This is partly due to inexperience but it is also due to analysts not understanding the methodology or technique that the company may be implementing at the same time that the Case tool is being adopted. Thus, one should not overlook the learning overhead for both the CASE technology and the various methodologies such as Gane & Sarson that the installation might be installing.

The human factor plays an important role here as it did when structured walkthroughs were first introduced 15 years ago. People are fearful of their jobs but, even if they overlook this factor, they may not trust the tools. Beyond this the tool and the learning curve associated with its use, can be used to cover up unsound design(s).

The interface of the various CASE tools both with other CASE tools and with other software packages that a given organization might use or plan on using such as a DBMS or 4GL is of vital concern as one contemplates the role of CASE in an organization. The articles by Myers and Weitz and the articles referenced earlier that develop evaluation criteria address these issues.

The next steps for CASE technology seem to be well defined: there will undoubtedly be a vendor shakeout, the major players will be identified, de-facto standards will be set, an increasing number of interfaces will be built between upper, mid and lower case tools, as well as, with other software. From a capability perspective, a new generation of knowledge-based expert systems will vault CASE into the hoped for productivity gains.

BIBLIOGRAPHY

Aranow, Eric. "When is CASE the Right Choice." Business Software Review. April 1988, pp.14-17.

Burkhard, Donald L. "Implementing CASE Tools." Journal of Systems Managenment. no.5, 1989, pp.20-25.

Danziger, Michael R. and Haynes, Phillip S. "Managing the CASE Environment." Journal of Systems Management. no.5, 1989,pp.29-32.

Gibson, Michael L. "A Guide to Selecting CASE Tools." Datamation, July 1, 1988,pp.65-66.

Hazzah, Ali. "Improvements in Tools Leading to Second Look." Software Magazine, September 1989, pp.31-47.

Knight, Robert. "CASE Payback Perceived, If Not Exactly Measured." Software News. February 1987, pp.56-58.

Leavitt, Don. "Scratching the Surface CASE's Potential." Software News, February 1987, pp.50-51.

McFadden, Fred and Discenza, Richard. "Confronting the Software Crisis." Business Horizons. Nov./Dec., 1987, pp.68-73.

Martin, Charles. F. "Second Generation CASE Tools: A Challenge to Vendors." Paper #17 from the First International Workshop on CASE. Index Technology Corp. Cambridge,MA.

Merlyn, Vaughan and Boone, Greg. "CASE TOOLS: Sorting out the tangle and tool types." COMPUTERWORLD, March 27, 1989, pp.65-70.

Myers, Edith. "Back-End Tools Seek Front-End Companions." Software Magazine, September 1988, pp.49-68.

Necco, Charles R., Tsai, Nancy W., and Holgeson, Kreg W. "Current Usage of CASE Software." Journal of Systems Management, no.5. 1989, pp.6-11.

Santosus, Cathleen. "The Fine Art of Figuring CASE Payback." Computerworld, March 27,1989,pp.74-75.

Troy, Douglas A. "An evaluation of CASE Tools." IEEE Proceedings of the Computer Software & Applications Conference 1987,pp.124-125.

Weitz, Lori. "Not Cannibalism, But Coexistence with CASE." Software Magazine, October 1988, pp.43-59.

Zucconi, L. "Selecting a CASE Tool." ACM Software Engineering Notes, Vol.14, no. 2, April 1989, pp.42-44.